CLASSIFYING MINIMAL VANISHING SUMS OF ROOTS OF UNITY

LOUIS CHRISTIE, KENNETH J. DYKEMA*, AND IGOR KLEP*

ABSTRACT. A vanishing sum of roots of unity is called minimal if no proper, nonempty sub-sum of it vanishes. This paper classifies all minimal vanishing sums of roots of unity of weight ≤ 16 by hand and then develops and codes an algorithm to extend the classification further, up to weight ≤ 21.

1. INTRODUCTION

A *root of unity* is a complex number ω, some positive integer power of which is equal to 1. This paper studies vanishing sums of roots of unity, i.e., integer combinations

$$\sum_{i=1}^{k} a_i \omega_i, \tag{1}$$

where $a_i \in \mathbb{Z}$ and ω_i are distinct roots of unity, so that the value of the sum is 0. If each proper sub-sum of such a sum is nonzero, we call (1) *minimal*. The *weight* of an integer combination of the form (1) is defined to be ℓ_1-norm of the coefficient vector, i.e., $\sum_i |a_i|$. The *height* is the ℓ_∞-norm, namely, $\max_i |a_i|$.

Vanishing sums of roots of unity naturally arise in a number of areas in algebra [Sch64, Ste08, AS12], geometry and combinatorics [PR98], and number theory [CJ76, DZ00, Len79, Man65]; we refer the reader to [LL00] for more details and further references. However, they are ubiquitous across many sciences. For instance, they recently appeared in operator algebra [JMS14], nuclear and particle physics [FG14] and approximation theory [CW11]. Mann [Man65] classifies all minimal vanishing sums of roots of unity of weight ≤ 7, Conway and Jones [CJ76] extend this to weight ≤ 9, and Poonen and Rubinstein [PR98] classify the ways in which 12 roots of unity can sum to zero. They introduced a notion of *type* of a minimal vanishing sum of roots of unity.

Our detailed investigations of vanishing sums of roots of unity grew out of an attempt to prove the Kaplansky-Lvov conjecture in degree 3 [DK20]. We needed a catalog of the minimal vanishing sums of roots of unity. It sufficed, for our purpose, to understand their types and their *parities*, namely, the number of roots of unity of odd and even order in the vanishing sum. Thus, in this paper, we extend the classification of Poonen and

Rubenstein [PR98] by extending the definition of type and cataloging the types and parities of all minimal vanishing sums of roots of unity of weight ≤ 21.

1.1. **Contributions and reader's guide.** Section 2 fixes notation and terminology, and presents a few of the preliminaries needed later in the paper. Our first main result, Theorem 3.3 in Section 3, classifies all such types and parities for all minimal vanishing sums of roots of unity of weight ≤ 16. There are 76 types in this classification, and the number of types increases rapidly with weight. The proof of Theorem 3.3 does not rely on use of computers. New phenomena arise when extending the classification beyond weight 12, see Proposition 2.3, Definition 2.4 and Subsection 2.2 for details. Our second main result identifies the lowest weight minimal vanishing sums of roots of unity of height > 1, which occur at weight 21; see Theorem 4.3 or Table 2. The large number of cases to handle forces us to turn to computer assistance, so in Section 4 we present an algorithm to find all minimal vanishing sums of a given weight; its implementation is given in Appendix B and the obtained list for weights ≤ 21 is given in Appendix A.

2. Notation and Preliminaries

For $n \in \mathbb{N}$, we write $\nu_n = e^{2\pi i/n}$ for the standard primitive n-th root of unity. The *order* of a root of unity ω is the least positive integer d such that $\omega^d = 1$.

A *sum of roots of unity* (also written *sorou*) is an unordered, finite, nonempty list $h = (\omega_1, \omega_2, \ldots, \omega_n)$ of roots of unity.[1] The *terms* of h are the roots of unity $\omega_1, \ldots, \omega_n$. The *value* of h is the complex number

$$\mathrm{val}(h) = \sum_{j=1}^{n} \omega_j$$

and the *weight* of h is the integer n. The *rotation* of h by the root of unity z is $zh = (z\omega_1, \ldots, z\omega_n)$. We write $h \sim zh$ and note that $\sim$ is an equivalence relation. The *order* of h is the least common multiple of the orders of the terms ω_j of h, and the *relative order* of h is the least common multiple of the orders of all the ratios ω_i/ω_j of all the terms of h. Note that a sorou with relative order d can always be rotated to obtain a sorou of order d.

As is traditional, we will usually write $\omega_1 + \cdots + \omega_n$ for the sorou, instead of writing it as a list. With this convention, the sorou $(1, -1)$ is represented as $1 + (-1)$, which is not the same as the complex number 0, though its value is, of course 0. As is natural with this notation, for sorou f and g we let $f + g$ denote the sorou that, technically, is the concatenation of the lists of terms of f and of g. The *multiplicity* of a root of unity z in a sorou h as above is the number of times it appears in h, namely, the number of $j \in \{1, \ldots, n\}$ such that $\omega_j = z$. We may write a sorou as a sum employing multiplicities. Thus, $2\nu_3 + \nu_3^2 + 3(-1)$ represents the sorou $h = (\nu_3, \nu_3, \nu_3^2, -1, -1, -1)$. The *height* of h is the maximum multiplicity of any of the ω_j that appears in h. We also employ natural conventions regarding minus signs, so that the same sorou h may also be written $2\nu_3 + \nu_3^2 - 3$, and $f - g$ means the sorou $f + (-1)g$.

A *subsorou* of a sorou $h = (\omega_1, \ldots, \omega_n)$ is a sorou g that can be written as $g = (\omega_{k(1)}, \omega_{k(2)}, \ldots, \omega_{k(m)})$ for some $1 \leq k(1) < \cdots < k(m) \leq n$, for some $m \in \{0, \ldots, n\}$.

---

[1] The plural of sorou is *sums of roots of unity*, which is also written as *sorou*.

If $m = 0$, then we say g is the *empty subsorou*, and if $m < n$, i.e., $g \neq h$, then we say g is a *proper subsorou*. We write $g \prec h$ when g is a subsorou of h.

A sorou is said to *vanish* if its value is 0. A vanishing sorou is said to be *minimal vanishing* if no proper, nonempty subsorou of it is vanishing. It is clear that every vanishing sorou can be written as a sum (or equivalently, in the list notation, as a concatenation) of minimal vanishing sorou, though not necessarily in a unique way.

As mentioned in the introduction, vanishing sorou have been studied by many authors. We will classify the minimal vanishing sorou of weight ≤ 21. This continues and extends the work [PR98] of Poonen and Rubenstein, who classified the minimal vanishing sorou of weight no more than 12.

The following lemma is a consequence of [Man65, Theorem 1].

Lemma 2.1. *The relative order of a minimal vanishing sorou h is a product $p_1 p_2 \cdots p_s$ of distinct primes $p_1 < p_2 < \cdots < p_s$.*

Definition 2.2. In the setting of the above lemma, we call p_s the *top prime* of h.

Proposition 2.3. *Let h be a sorou whose order is a product $p_1 p_2 \ldots p_s$ of distinct primes $p_1 < p_2 < \cdots < p_s$. Let $p = p_s$ be the top prime. Then, after replacing h by a rotation, if necessary, we have*

$$h = \sum_{j=0}^{p-1} \nu_p^j f_j, \tag{2}$$

for some sorou $f_0, \ldots, f_{p-1}$, each term of which has order dividing $p_1 p_2 \cdots p_{s-1}$. Then h vanishes if and only if

$$\mathrm{val}(f_0) = \mathrm{val}(f_1) = \cdots = \mathrm{val}(f_{p-1}). \tag{3}$$

Suppose that h vanishes. Then h is minimal vanishing if and only if the following hold:

(i) $\mathrm{val}(f_0) \neq 0$,
(ii) *for no j does f_j have a vanishing proper, nonempty, subsorou,*
(iii) *there is no complex number z such that for all j, f_j has a proper, nonempty subsorou with value z.*

Proof. The first statement follows from the fact that $\mathbb{Q}(\nu_p, \nu_{p_{s-1}} \cdots \nu_{p_1})/\mathbb{Q}(\nu_{p_{s-1}} \cdots \nu_{p_1})$ is a field extension of degree p, and the minimal polynomial for ν_p is $\Phi_p(x) = \sum_{j=0}^{p-1} x^j$ (cf. [Was97, Proposition 2.4]). Then the minimality characterization follows easily. ∎

Note that equation (3) is equivalent to the fact that for all $i, j \in \{0, \ldots, p-1\}$, the sorou $f_i - f_j$ vanishes.

From Proposition 2.3, we can deduce the well known fact (cf. Definition 2.4(e) below) that if h is a minimal vanishing sorou of prime order p, then up to rotation h equals

$$1 + \nu_p + \nu_p^2 \cdots + \nu_p^{p-1}.$$

Following [PR98], we say that such a vanishing sorou has type R_p.

2.1. **Types and parities.**

Definition 2.4. We recursively associate to an arbitrary vanishing sorou h (or, more correctly, to the equivalence class of h under rotation) its *type* (or types) as follows.

(a) If h is a sum of minimal vanishing sorou $v_1, \ldots, v_k$ having types $T_1, \ldots, T_k$, respectively, then we say h has type $T_1 \oplus T_2 \oplus \cdots \oplus T_k$.

(b) A minimal vanishing sorou h, after rotation, can always be written as in (2) with $w(f_0) \le w(f_j)$ for all $j \le p-1$ and and with $1 \prec f_0$. We call $f_0, \ldots, f_{p-1}$ the *subsidiary sorou* of h.

(c) The weights $w(f_0), \ldots, w(f_{p-1})$, when arranged in increasing order, form the *subsidiary weight partition* of h, often called just the *weight partition* of h.

(d) We call f_0 a *smallest weight subsidiary sorou* of h. Note that f_0 is not, in general, unique, even up to rotation, since we allow rotations of h, but the weight of f_0 is unique, and is called the *smallest subsidiary weight* of h.

(e) If $f_j = f_0$ for all j, then f_0 can have only one term, namely, 1, (for otherwise, by Proposition 2.3, h would fail to be minimal vanishing); in this case, as noted already, we say f has type R_p.

(f) For minimal vanishing h written (after rotation) as in (b), let J be the set of all $j \in \{1, \ldots, p-1\}$ for which $f_j \ne f_0$ and suppose $J = \{j(1), j(2), \ldots, j(n)\}$ is nonempty. Suppose the vanishing sorou $f_0 - f_{j(i)}$ has type T_i. Then we say that h has type

$$(R_p : f_0 : T_1, T_2, \ldots, T_n), \tag{4}$$

where the ordering of $T_1, \ldots, T_n$ is unimportant. We call $T_1, \ldots, T_n$ the *subsidiary types* of h.

(g) In the case $f_0 = 1$, we may omit the "f_0 :", writing instead

$$(R_p : T_1, T_2, \ldots, T_n).$$

Furthermore, whenever some of the subsidiary types T_i are repeated, the multiplicity may be indicated with an integer. For example, $(R_7 : 3R_3, 2R_5)$ instead of $(R_7 : R_3, R_3, R_3, R_5, R_5)$.

Remark 2.5.

(a) The above definition builds upon the definition used in [PR98] (which was confined to the case $f_0 = 1$).

(b) A given vanishing sorou may have more than one type. For example $1 + \nu_3 + \nu_3^2 - 1 - \nu_3 - \nu_3^2$ has type $R_3 \oplus R_3$ and type $R_2 \oplus R_2 \oplus R_2$.

(c) In some cases, for example R_p or $(R_p : R_q)$, the type specifies the sorou uniquely up to rotation. But not in other cases, for example $(R_5 : 2R_3)$.

(d) The types of minimal vanishing sorou are distinct from the types of non-minimal vanishing sorou.

(e) If a minimal vanishing sorou has type $(R_p : T_1, \ldots, T_n)$, namely, with smallest subsidiary weight 1, then all of the types T_i must be of minimal vanishing sorou.

(f) More generally, if the type is $(R_p : f_0 : T_1, \ldots, T_n)$, then each T_j decomposes as a sum of at most $w(f_0)$ minimal vanishing sorou (for otherwise, f_j would itself have a proper, nonempty, vanishing subsorou).

We also define the *parity* of a sorou as the pair of integers counting the number of positive and negative signs in the sorou. Note that because we allow rotations, the order of the pair is irrelevant.

2.2. **A few examples.** Now that we have introduced basic terminology, let us illustrate it with a few examples. Consider the sorou $h_1 = 1 + \nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4$ of type R_5 and the sorou $h_2 = 1 + \nu_3 + \nu_3^2$ of type R_3. Then $h_1' = \nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4$ and $h_2' = \nu_3 + \nu_3^2$ are both sorou having valuation -1, and subtracting them yields the minimal vanishing sorou

$$h = h_1' - h_2' = \nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4 - \nu_3 - \nu_3^2,$$

which can be rewritten

$$h = \nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4 + \nu_6 + \nu_6^5 \sim \sum_{j=0}^{4} f_j \nu_5^j$$

where $f_0 = f_1 = f_2 = f_3 = 1$ and $f_4 = \nu_6 + \nu_6^5$. Thus $f_0 - f_4 = 1 - \nu_6 - \nu_6^5 = 1 + \nu_3 + \nu_3^2$ and so we have a minimal vanishing sorou, with type $(R_5 : R_3)$. In a similar manner, we obtain

$$g = \nu_5^2 + \nu_5^3 + \nu_5^4 - \nu_3 - \nu_3^2 - \nu_5\nu_3 - \nu_5\nu_3^2$$

of type $(R_5 : 2R_3)$, etc. These are pictured below.

R_5	h of type $(R_5 : R_3)$	g of type $(R_5 : 2R_3)$
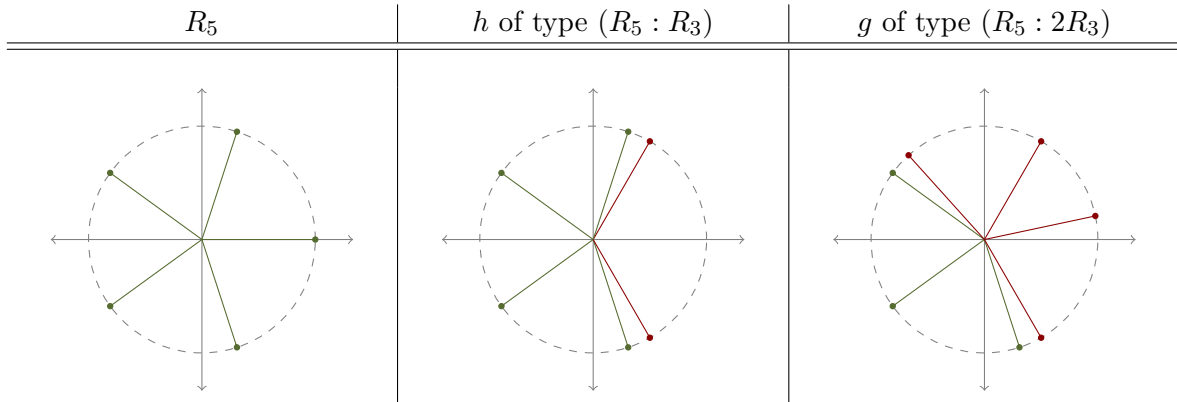
Now consider the minimal vanishing sorou $\sum_{j=0}^{6} \nu_7^j$. Multiplying by $1 + \nu_5$ yields another vanishing, but not minimal sorou, but then subtracting h_1 gives:

$$\left(\sum_{i=0}^{6} \nu_7^i\right)(1 + \nu_5) - \left(\sum_{i=0}^{4} \nu_5^j\right) \sim \sum_{j=0}^{6} f_j \nu_7^j,$$

where

$$f_0 = f_1 = \cdots = f_5 = 1 + \nu_5 \qquad \text{and} \qquad f_6 = \nu_{10} + \nu_{10}^3 + \nu_{10}^9,$$

a minimal vanishing sorou of type $(R_7 : 1 + \nu_5 : R_5)$, with $f_0 \succ 1$. For types with $f_0 \succ 1$ we can have non-minimal subtypes (see Remark 2.5(f)) as depicted below.

$(R_7 : 1 + \nu_5 : R_5)$	$(R_7 : 1 + \nu_{15}^2 : (R_3 \oplus R_5), (R_5 : 2R_3))$
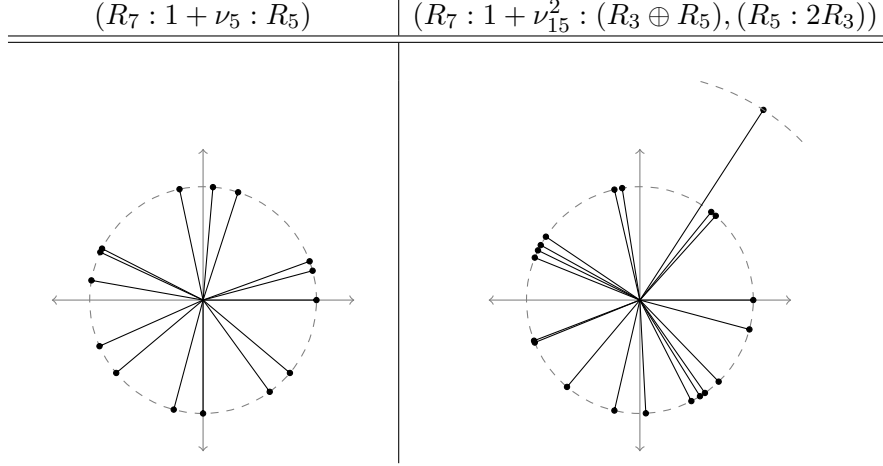
In fact, the minimal vanishing sorou $h = \sum_{j=0}^{6} \nu_7^j f_j$ shown above with

$$f_j = 1 + \nu_3 \nu_5^4 \quad (j \in \{0, 1, 2, 3, 5\})$$
$$f_4 = -\nu_3 - \nu_5^4 + \nu_3^2(\nu_5 + \nu_5^2 + \nu_5^3)$$
$$f_6 = -\nu_5 - \nu_5^2 - \nu_5^3 - 2\nu_5^4 - \nu_3^2\nu_5^4$$

of height 2 and weight 21 is of type $(R_7 : 1 + \nu_{15}^2 : (R_3 \oplus R_5), (R_5 : 2R_3))$ and, as is shown later, this is one of only five types of minimal vanishing sorou having height > 1 and weight ≤ 21.

Verification that h is minimal vanishing using the criteria of Proposition 2.3 can be done "by hand", symbolically or numerically using both Mathematica and our python code.

3. Classification of sorou of weight ≤ 16

The main result of this section is Theorem 3.3 that classifies minimal vanishing sorou of weight ≤ 16. Theorem 3.2 presents a simple characterization of minimal vanishing sorou whose relative order divides $2pq$.

The following is Lemma 3.3 of [PR98] (and an immediate consequence of it).

Lemma 3.1. *Let p be an odd prime. The only minimal vanishing sorou of relative order dividing $2p$ are of types R_2 and R_p. Thus, the only vanishing sorou of relative order dividing $2p$ are of types $R_2^{\oplus m} \oplus R_p^{\oplus n}$ for $n, m \geq 0$.*

Theorem 3.2. *Let $p < q$ be odd primes. Suppose h is a minimal vanishing sorou whose relative order divides $2pq$. Then either h is of type R_2, R_p or R_q or there are proper, nonempty subsets I of $\{0, 1, \ldots, p-1\}$ and J of $\{0, 1, \ldots, q-1\}$ such that $0 \in I$, I has cardinality no more than $(p-1)/2$ and such that a rotation of h is equal to the sorou*

$$\sum_{j \in J^c} \left(\sum_{i \in I} \nu_p^i \right) \nu_q^j + \sum_{j \in J} \left(\sum_{i \in I^c} (-\nu_p^i) \right) \nu_q^j,$$

where $I^c = \{0, 1, \ldots, p-1\} \setminus I$ and $J^c = \{0, 1, \ldots, q-1\} \setminus J$. Thus, h has type

$$(R_q : \sum_{i \in I} \nu_p^i : |J|R_p).$$

Proof. We may suppose h is not of type R_2, R_p of R_q. Thus, (by Lemma 3.1) its top prime is q and, after rotating h, if necessary, we may write

$$h = \sum_{j=0}^{q-1} f_j \nu_q^j,$$

where each f_j is a sum of $2p$-th roots of unity, $\mathrm{val}(f_j) \neq 0$ is independent of j, $1 \prec f_0$ and $w(f_0) \leq w(f_j)$ for all j. Let a_i be the multiplicity of ν_p^i in f_0 and b_i the multiplicity of $-\nu_p^i$ in f_0. Thus,

$$f_0 = \sum_{i=0}^{p-1} a_i \nu_p^i + \sum_{i=0}^{p-1} b_i(-\nu_p^i).$$

Since f_0 has no nonempty vanishing subsorou, we must have $a_i b_i = 0$ for all i. Furthermore, letting $I = \{i \in \{0, \ldots, p-1\} \mid a_i > 0\}$, we have $0 \in I$ and I must be a proper, nonempty subset of $\{0, \ldots, p-1\}$, for otherwise f_0 would have a vanishing subsum of type R_p.

Let $j \in \{0, \ldots, 1-q\}$. Since $f_0 - f_j$ is a vanishing sum of $2p$-th roots of unity, by Lemma 3.1, it has type $R_2^{\oplus k(j)} \oplus R_p^{\oplus \ell(j)}$ for some $k(j), \ell(j) \geq 0$. Though the pair $(k(j), \ell(j))$ is not necessarily unique, we fix a choice, for each j. Moreover, we may (and do) choose $k(0) = w(f_0)$ and $\ell(0) = 0$. We also fix, for each j, a decomposition of $f_0 - f_j$ into a sum of $k(j)$ minimal vanishing subsorou of type R_2 and $\ell(j)$ minimal vanishing subsorou of type R_p and thereby associate, to each term of f_0, the corresponding subsorou of $f_0 - f_j$ (either of type R_2 or of type R_p) in which that term appears. Of the a_i terms of f_0 that are equal to ν_p^i, let $c_i^{(j)}$ be the number of them whose associated subsorou (as above) of $f_0 - f_j$ is of type R_p. Similarly, of the b_i terms of f_0 that are equal to $-\nu_p^i$, let $d_i^{(j)}$ be the number of them whose associated subsorou of $f_0 - f_j$ is of type R_p. Note that we have

$$0 \leq c_i^{(j)} \leq a_i, \qquad 0 \leq d_i^{(j)} \leq b_i. \tag{5}$$

Each of the $\ell(j)$ minimal vanishing subsorou of $f_0 - f_j$ of type R_p in the fixed decomposition of $f_0 - f_j$ must be of the form $1 + \nu_p + \nu_p^2 + \cdots + \nu_p^{p-1}$ or $-1 - \nu_p - \nu_p^2 - \cdots - \nu_p^{p-1}$. Let $\ell_+(j)$ and $\ell_-(j)$ be the number of each form, respectively. Thus, for all $j \in \{0, \ldots, q-1\}$, we have

$$\ell_+(j) \geq \max_{0 \leq i \leq p-1} c_i^{(j)}, \qquad \ell_-(j) \geq \max_{0 \leq i \leq p-1} d_i^{(j)}. \tag{6}$$

We now consider the multiplicity of ν_p^i in $-f_j$. There are $b_i - d_i^{(j)}$ terms in f_0 equal to $-\nu_p^i$ that appear in one of the $k(j)$ vanishing subsorou of $f_0 - f_j$ having type R_2 in the fixed decomposition. These contribute $b_i - d_i^{(j)}$ to the multiplicity of ν_p^i in $-f_j$. The other appearances of ν_p^i in f_j occur in those of the $\ell_+(j)$ subsorou of $f_0 - f_j$ of the form $1 + \nu_p + \cdots + \nu_p^{p-1}$ in the fixed decomposition in which the term ν_p^i does not come from f_0. There are $\ell_+(j) - c_i^{(j)}$ such appearances. Thus, the multiplicity of ν_p^i in $-f_j$ is equal to $b_i - d_i^{(j)} + \ell_+(j) - c_i^{(j)}$. Similarly, the multiplicity of $-\nu_p^i$ in $-f_j$ is $a_i - c_i^{(j)} + \ell_-(j) - d_i^{(j)}$. Thus, we have

$$f_j = \sum_{i=0}^{p-1} \left(a_i - c_i^{(j)} + \ell_-(j) - d_i^{(j)}\right) \nu_p^i + \sum_{i=1}^{p-1} \left(b_i - d_i^{(j)} + \ell_+(j) - c_i^{(j)}\right)(-\nu_p^i).$$

If $\ell_+(j) = 0$, then, by (6), $c_i^{(j)} = 0$ for all i. Then, for all $i \in I$, we have

$$a_i - c_i^{(j)} + \ell_-(j) - d_i^{(j)} \geq a_i > 0.$$

Consequently, we have

$$\sum_{i \in I} \nu_p^i \prec f_j. \tag{7}$$

If $\ell_+(j) > 0$, then since, for all $i \in I^c$ we have $a_i = 0$, by (5) we have $c_i^{(j)} = 0$ and, thus,

$$b_i - d_i^{(j)} + \ell_+(j) - c_i^{(j)} \geq \ell_+(j) > 0.$$

Consequently, we have

$$\sum_{i \in I^c} (-\nu_p^i) \prec f_j. \tag{8}$$

Let $J = \{j \in \{0, \ldots, q-1\} \mid \ell_+(j) > 0\}$. Since $\ell(0) = 0$, we have $0 \notin J$. Using (7) and (8), we have

$$\sum_{j \in J^c} \left(\sum_{i \in I} \nu_p^i \right) \nu_q^j + \sum_{j \in J} \left(\sum_{i \in I^c} (-\nu_p^i) \right) \nu_q^j \prec h. \tag{9}$$

Since

$$\mathrm{val} \left(\sum_{i \in I} \nu_p^i \right) = \mathrm{val} \left(\sum_{i \in I^c} (-\nu_p^i) \right),$$

the sorou on the left-hand-side of (9) vanishes. Since h is a minimal vanishing sorou, we must have equality in (9). The set J cannot be empty, for this would violate that h is minimal vanishing. Finally, by suitable rotation, the roles of I and I^c can be interchanged, simultaneously with those of J and J^c. Thus, after rotation if necessary, we may obtain that the cardinality of I is no greater than $(p-1)/2$.   ∎

The following theorem extends the work of Poonen and Rubenstein [PR98], whose classification went up to weight 12.

Theorem 3.3. *Types of all of the minimal vanishing sums of roots of unity of weight no greater than 16, up to rotation, are listed in Table 2. All have height 1. Also listed (to help with the derivation) are the top prime and the weight partition of each type. Furthermore, the possible parities of orders of terms of the sorou of each type are listed. All of the indicated possible parities do, in fact, occur.*

Table 1: Minimal vanishing sums of roots of unity and their possible
parities of orders.

Weight	Top prime	Relative order	Weight partition	Type	Possible parities
2	2	2	$(1,1)$	R_2	$(1,1)$
3	3	3	$(1,1,1)$	R_3	$(3,0)$
5	5	5	$(1,1,1,1,1)$	R_5	$(5,0)$
6	5	30	$(1,1,1,1,2)$	$(R_5 : R_3)$	$(4,2)$
7	7	7	$(1,1,1,1,1,1,1)$	R_7	$(7,0)$
	5	30	$(1,1,1,2,2)$	$(R_5 : 2R_3)$	$(4,3)$

Continued on next page

Table 1 – *Continued from previous page*

weight	prime	order	partition	type	parities
8	7	42	$(1,1,1,1,1,1,2)$	$(R_7 : R_3)$	$(6,2)$
	5	30	$(1,1,2,2,2)$	$(R_5 : 3R_3)$	$(6,2)$
9	7	42	$(1,1,1,1,1,2,2)$	$(R_7 : 2R_3)$	$(5,4)$
	5	30	$(1,2,2,2,2)$	$(R_5 : 4R_3)$	$(8,1)$
10	7	70	$(1,1,1,1,1,1,4)$	$(R_7 : R_5)$	$(6,4)$
	7	42	$(1,1,1,1,2,2,2)$	$(R_7 : 3R_3)$	$(6,4)$
11	11	11	$(1,\dots,1)$	R_{11}	$(11,0)$
	7	210	$(1,1,1,1,1,1,5)$	$(R_7 : (R_5 : R_3))$	$(10,1)$, $(8,3)$
	7	210	$(1,1,1,1,1,2,4)$	$(R_7 : R_3, R_5)$	$(6,5)$
	7	42	$(1,1,1,2,2,2,2)$	$(R_7 : 4R_3)$	$(8,3)$
12	11	66	$(1,\dots,1,2)$	$(R_{11} : R_3)$	$(10,2)$
	7	210	$(1,1,1,1,1,1,6)$	$(R_7 : (R_5 : 2R_3))$	$(10,2)$, $(9,3)$
	7	210	$(1,1,1,1,1,2,5)$	$(R_7 : R_3, (R_5 : R_3))$	$(9,3)$, $(7,5)$
	7	210	$(1,1,1,1,2,2,4)$	$(R_7 : 2R_3, R_5)$	$(8,4)$
	7	42	$(1,1,2,2,2,2,2)$	$(R_7 : 5R_3)$	$(10,2)$
13	13	13	$(1,\dots,1)$	R_{13}	$(13,0)$
	11	66	$(1,\dots,1,2,2)$	$(R_{11} : 2R_3)$	$(9,4)$
	7	210	$(1,1,1,1,1,1,7)$	$(R_7 : (R_5 : 3R_3))$	$(12,1)$, $(8,5)$
	7	210	$(1,1,1,1,1,2,6)$	$(R_7 : R_3, (R_5 : 2R_3))$	$(9,4)$, $(8,5)$
	7	70	$(1,1,1,1,1,4,4)$	$(R_7 : 2R_5)$	$(8,5)$
	7	210	$(1,1,1,1,2,2,5)$	$(R_7 : 2R_3, (R_5 : R_3))$	$(8,5)$, $(7,6)$
	7	210	$(1,1,1,2,2,2,4)$	$(R_7 : 3R_3, R_5)$	$(10,3)$
	7	42	$(1,2,2,2,2,2,2)$	$(R_7 : 6R_3)$	$(12,1)$
14	13	78	$(1,\dots,1,2)$	$(R_{13} : R_3)$	$(12,2)$
	11	110	$(1,\dots,1,4)$	$(R_{11} : R_5)$	$(10,4)$
	11	66	$(1,\dots,1,2,2,2)$	$(R_{11} : 3R_3)$	$(8,6)$
	7	210	$(1,1,1,1,1,1,8)$	$(R_7 : (R_5 : 4R_3))$	$(14,0)$, $(7,7)$
	7	210	$(1,1,1,1,1,2,7)$	$(R_7 : R_3, (R_5 : 3R_3))$	$(11,3)$, $(7,7)$
	7	210	$(1,1,1,1,1,4,5)$	$(R_7 : R_5, (R_5 : R_3))$	$(9,5)$, $(7,7)$
	7	210	$(1,1,1,1,2,2,6)$	$(R_7 : 2R_3, (R_5 : 2R_3))$	$(8,6)$, $(7,7)$
	7	210	$(1,1,1,1,2,4,4)$	$(R_7 : R_3, 2R_5)$	$(10,4)$
	7	210	$(1,1,1,2,2,2,5)$	$(R_7 : 3R_3, (R_5 : R_3))$	$(9,5)$, $(7,7)$
	7	210	$(1,1,2,2,2,2,4)$	$(R_7 : 4R_3, R_5)$	$(12,2)$
15	13	78	$(1,\dots,1,2,2)$	$(R_{13} : 2R_3)$	$(11,4)$
	11	330	$(1,\dots,1,5)$	$(R_{11} : (R_5 : R_3))$	$(14,1)$, $(12,3)$
	11	330	$(1,\dots,1,2,4)$	$(R_{11} : R_3, R_5)$	$(9,6)$
	11	66	$(1,\dots,1,2,2,2,2)$	$(R_{11} : 4R_3)$	$(8,7)$
	7	210	$(1,1,1,1,1,2,8)$	$(R_7 : R_3, (R_5 : 4R_3))$	$(13,2)$, $(9,6)$
	7	210	$(1,1,1,1,1,4,6)$	$(R_7 : R_5, (R_5 : 2R_3))$	$(9,6)$, $(8,7)$
	7	210	$(1,1,1,1,1,5,5)$	$(R_7 : 2(R_5 : R_3))$	$(13,2)$, $(11,4)$, $(9,6)$
	7	210	$(1,1,1,1,2,2,7)$	$(R_7 : 2R_3, (R_5 : 3R_3))$	$(10,5)$, $(9,6)$
	7	210	$(1,1,1,1,2,4,5)$	$(R_7 : R_3, R_5, (R_5 : R_3))$	$(9,6)$, $(8,7)$
	7	210	$(1,1,1,2,2,2,6)$	$(R_7 : 3R_3, (R_5 : 2R_3))$	$(9,6)$, $(8,7)$
	7	210	$(1,1,1,2,2,4,4)$	$(R_7 : 2R_3, 2R_5)$	$(12,3)$
	7	210	$(1,1,2,2,2,2,5)$	$(R_7 : 4R_3, (R_5 : R_3))$	$(11,4)$, $(9,6)$
	7	210	$(1,2,2,2,2,2,4)$	$(R_7 : 5R_3, R_5)$	$(14,1)$
	7	70	$(2,2,2,2,2,2,3)$	$(R_7 : 1 + \nu_5^y : R_5)$ $y \in \{1,2\}$	$(12,3)$
16	13	130	$(1,\dots,1,4)$	$(R_{13} : R_5)$	$(12,4)$

Table 1 – *Continued from previous page*

weight	prime	order	partition	type	parities
	13	78	$(1,\ldots,1,2,2,2)$	$(R_{13}:3R_3)$	$(10,6)$
	11	154	$(1,\ldots,1,6)$	$(R_{11}:R_7)$	$(10,6)$
	11	330	$(1,\ldots,1,6)$	$(R_{11}:(R_5:2R_3))$	$(14,2),\,(13,3)$
	11	330	$(1,\ldots,1,2,5)$	$(R_{11}:R_3,(R_5:R_3))$	$(13,3),\,(11,5)$
	11	330	$(1,\ldots,1,2,2,4)$	$(R_{11}:2R_3,R_5)$	$(8,8)$
	11	66	$(1,\ldots,1,2,2,2,2,2)$	$(R_{11}:5R_3)$	$(10,6)$
	7	210	$(1,1,1,1,1,4,7)$	$(R_7:R_5,(R_5:3R_3))$	$(11,5),\,(9,7)$
	7	210	$(1,1,1,1,1,5,6)$	$(R_7:(R_5:R_3),(R_5:2R_3))$	$(13,3),\,(12,4)$ $(11,5),\,(10,6)$
	7	210	$(1,1,1,1,2,2,8)$	$(R_7:2R_3,(R_5:4R_3))$	$(12,4),\,(11,5)$
	7	210	$(1,1,1,1,2,4,6)$	$(R_7:R_3,R_5,(R_5:2R_3))$	$(9,7),\,(8,8)$
	7	210	$(1,1,1,1,2,5,5)$	$(R_7:R_3,2(R_5:R_3))$	$(12,4),\,(10,6),\,(8,8)$
	7	210	$(1,1,1,1,4,4,4)$	$(R_7:3R_5)$	$(12,4)$
	7	210	$(1,1,1,2,2,2,7)$	$(R_7:3R_3,(R_5:3R_3))$	$(11,5),\,(9,7)$
	7	210	$(1,1,1,2,2,4,5)$	$(R_7:2R_3,R_5,(R_5:R_3))$	$(11,5),\,(9,7)$
	7	210	$(1,1,2,2,2,2,6)$	$(R_7:4R_3,(R_5:2R_3))$	$(11,5),\,(10,6)$
	7	210	$(1,1,2,2,2,4,4)$	$(R_7:3R_3,2R_5)$	$(14,2)$
	7	210	$(1,2,2,2,2,2,5)$	$(R_7:5R_3,(R_5:R_3))$	$(13,3),\,(11,5)$
	7	105	$(2,2,2,2,2,2,4)$	$(R_7:1+\nu_3:(R_5:R_3))$	$(16,0)$
	7	210	$(2,2,2,2,2,2,4)$	$(R_7:1+\nu_5^y:(R_5:R_3))$ $y\in\{1,2\}$	$(14,2)$
	7	210	$(2,2,2,2,2,2,4)$	$(R_7:1-\nu_3\nu_5^y:(R_5:R_3))$ $y\in\{1,2,3,4\}$	$(9,7)$
	7	210	$(2,2,2,2,2,3,3)$	$(R_7:1+\nu_5^y:R_2\oplus R_3,\,R_5)$ $y\in\{1,2\}$	$(11,5)$
	7	70	$(2,2,2,2,2,3,3)$	$(R_7:1+\nu_5^y:2R_5)$ $y\in\{1,2\}$	$(10,6)$

Regarding parities, we have for example, that a minimal vanishing sorou of type $(R_5:R_3)$ can have either 4 terms of even order and 2 of odd order, or 4 of odd order and 2 of even order, while one of type $(R_7:(R_5,R_3))$ can have either 10 even and 1 odd, 8 even and 3 odd, 3 odd and 8 even or 1 odd and 10 even.

Proof of Theorem 3.3. First of all, note that, by Theorem 3.2, if the top prime of a minimal vanishing sorou is ≤ 5, then the weight partition must contain 1.

For those types whose weight partition contains 1, the cataloging is essentially a continuation of the method of Poonen and Rubenstein [PR98]. We will now describe the general technique in this case, while we work through two examples.

Suppose h is a minimal vanishing sorou with weight n. For example, say $n = 13$. Let p be the largest prime divisor of the relative order of h. For example, say $p = 7$. Write h as in equation (2) of Proposition 2.3. Then $(w(f_0),\ldots,w(f_{p-1}))$ is a partition of n into p parts, the nondecreasing re-ordering of which will be the weight partition. For this portion of the proof, we are supposing that 1 appears in this partition. After rotating h, if necessary, we may without loss of generality assume $f_0 = 1$. For example, suppose the weight partition is $(1,1,1,1,1,4,4)$. By Proposition 2.3, whenever $w(f_j) = 1$ but $j > 0$, we need $f_0 - f_j$ to be vanishing of weight 2, so we must have $f_j = 1$. If $w(f_j) > 1$, then we need $f_0 - f_j$ to

be vanishing. Moreover, since $f_0 = 1$ and f_j must have no vanishing nonempty subsorou, $f_0 - f_j$ must be a minimal non-vanishing sorou of weight $w(f_j) + 1$ and must have 1 as a subsorou and must have relative order that is a product of primes that are each strictly less than p. Thus, the construction of this portion of the table (when the weight partition contains 1) proceeds recursively. In the case of $p = 7$ and $w(f_j) = 4$, we must have $f_0 - f_j$ of type R_5. This forces $f_0 - f_j$ to have type R_5 and f_j to be $-\nu_5 - \nu_5^2 - \nu_5^3 - \nu_5^4$. This implies that the order parities are $(8, 5)$ — in this case, five odd order terms of the form ν_7^j for the j such that $f_j = 1$, and a total of eight even order terms from two sums of the form $\nu_7^j(-\nu_5 - \nu_5^2 - \nu_5^3 - \nu_5^4)$ for the j such that $w(f_j) = 4$. After rotation, of course, the odds and the evens can be interchanged. This is the reason we don't specify how many of each on the "Parities" column of Table 2. This shows that all minimal vanishing sorou with prime 7 and weight partition $(1, 1, 1, 1, 1, 4, 4)$ have parities $(8, 5)$. Conversely, using Proposition 2.3, we see that minimal vanishing sorou with this prime and weight partition do occur.

We now consider the case of weight partition $(1, 1, 1, 1, 2, 2, 5)$. Arguing as before, if h is a vanishing sorou with this weight partition, after rotating we have $f_0 = 1$ and, for those j with $w(f_j) = 1$ we have $f_j = 1$. For those j with $w(f_j) = 2$, since $f_0 - f_j$ is a minimal vanishing sorou, we need $f_j = -\nu_3 - \nu_3^2$. For j with $w(f_j) = 5$, $f_0 - f_j$ must have type (R_5, R_3), meaning it must be equal to a rotation of

$$1 + \nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4(-\nu_3 - \nu_3^2).$$

Thus, f_j must be one of the sorou

$$-\nu_5 - \nu_5^2 - \nu_5^3 + \nu_5^4(\nu_3 + \nu_3^2),$$
$$-\nu_5 - \nu_5^2 + \nu_5^3(\nu_3 + \nu_3^2) - \nu_5^4,$$
$$-\nu_5 + \nu_5^2(\nu_3 + \nu_3^2) - \nu_5^3 - \nu_5^4,$$
$$\nu_5(\nu_3 + \nu_3^2) - \nu_5^2 - \nu_5^3 - \nu_5^4,$$
$$-\nu_3 + \nu_3^2(\nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4)$$
$$\nu_3(\nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4) - \nu_3^2$$

Counting the terms with odd and, respectively, even orders, we get (unordered) parity distributions $(8, 5)$ and $(7, 6)$ for h, depending on which of the above six is chosen. Conversely, we see that each choice leads to a minimal vanishing sorou with the given weight partition, so that both parity distributions occur.

We now analyze the situations where the smallest subsidiary weight of a minimal vanishing sorou h is strictly greater than 1. By Theorem 3.2, this is possible only for top prime at least 7. We cannot have weight partition of the form $(2, 2, \ldots, 2)$, for this would require all subsidiary types to be $R_2 \oplus R_2$, which would imply, writing h as in (2), that $f_0 = f_1 = \cdots = f_{p-1}$ and then h would have a proper, nonempty vanishing subsorou of type R_p. More generally, if the smallest subsidiary weight is equal to 2, then assuming (without loss of generality) $1 \prec f_0$ and letting $(R_p : 1 + b : T_1, \ldots, T_n)$ be the type of h, at least one of the subsidiary types $T_1, \ldots, T_n$ must be minimal vanishing. Indeed if not, then we must be able to write each $f_0 - f_j = g_{j,1} + g_{j_2}$ for $g_{j,i}$ vanishing of type $T_{j,i}$, with $T = T_{j,1} \oplus T_{j,2}$. Then, after interchanging $g_{j,1}$ and $g_{j,2}$, if necessary, must have $1 \prec g_{j,1}$ and $b \prec g_{g,2}$, for

otherwise we would get a proper, vanishing subsorou $\nu_p^j g_{j,i}$ of h. But then, separating the bits, this implies that h has a proper, vanishing subsorou of type

$$(R_p : T_{1,1}, T_{2,1}, \ldots, T_{n,1}). \tag{10}$$

Thus, the lowest weight possibility is with top prime 7, weight partition $(2, 2, 2, 2, 2, 2, 3)$ and total weight 15. Suppose h is a minimal vanishing sorou with this data. Then

$$h = \sum_{j=0}^{6} \nu_7^j f_j \tag{11}$$

and after rotation, we may assume $1 \prec f_0$, $w(f_0) = w(f_1) = \cdots = w(f_5) = 2$ and $w(f_6) = 3$. Hence, $f_0 = 1 + b$ with b a root of unity whose order divides 30. For $1 \le j \le 5$, $f_0 - f_j$ must have type $R_2 \oplus R_2$ and, therefore, we have $f_j = f_0$. The type of $f_0 - f_6$ is either $R_2 \oplus R_3$ or R_5. Since, by the earlier remark, at least one of the subsidiary types must be minimal vanishing, the type of $f_0 - f_6$ must be R_5. After rotation, we must have $b \in \{\nu_5, \nu_5^2\}$. These yield the two minimal vanishing sorou

$$\left(\sum_{j=0}^{6} \nu_7^j (1 + \nu_5) \right) + \nu_7^6(-\nu_5^2 - \nu_5^3 - \nu_5^4)$$

$$\left(\sum_{j=0}^{6} \nu_7^j (1 + \nu_5^2) \right) + \nu_7^6(-\nu_5 - \nu_5^3 - \nu_5^4),$$

having height 1 and parities $(12, 3)$. These are of order $2 \cdot 5 \cdot 7$ and were, in fact, treated in Theorem 3.2.

We now turn to the case of top prime $p = 7$ and weight partition $(2, 2, 2, 2, 2, 2, 4)$, for total weight 16. Arguing as before, after rotation we must have h as in (11) with $f_0 = f_1 = \cdots = f_5 = 1 + b$ and with $f_0 - f_6$ vanishing of weight 6. By the observation made above (10), $f_0 - f_6$ must be minimal vanishing of weight 6, so must be of type $(R_5 : R_3)$. Since $1 \prec f_0$, after rotation we must have $f_0 = 1 + \nu_5^y$, some $y \in \{1, 2\}$, or $f_0 = 1 + \nu_3$ or $f_0 = 1 - \nu_3\nu_5^y$, some $y \in \{1, 2, 3, 4\}$. In the first case, when $f_0 = 1 + \nu_5^y$, there are three different choices for f_6, but they all yield minimal vanishing sorou of height 1 and parities $(14, 2)$. For example, when $f_0 = 1 + \nu_5$, then we must have either

$$h = \sum_{j=0}^{5} \nu_7^j (1 + \nu_5) + \nu_7^6(-\nu_5^2 - \nu_5^3 + \nu_5^4(\nu_3 + \nu_3^2))$$

or one of two other similar possibilities. When $f_0 = 1 + \nu_3$, then we must have $f_6 = \nu_3^2(\nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4)$, and we get

$$h = \sum_{j=0}^{5} \nu_7^j (1 + \nu_3) + \nu_7^6 \nu_3^2 (\nu_5 + \nu_5^2 + \nu_5^3 + \nu_5^4)$$

with height 1 and parities $(16, 0)$. The third case yields h with height 1 and parities $(9, 7)$. For example if $f_0 = 1 - \nu_3\nu_5$, the f_6 must be one of

$$\nu_3^2\nu_5 - \nu_5^2 - \nu_5^3 - \nu_5^4 \quad \text{or} \quad -\nu_3^2 + \nu_3\nu_5^2 + \nu_3\nu_5^3 + \nu_3\nu_5^4,$$

yielding

$$h = \sum_{j=0}^{5} \nu_7^j (1 - \nu_3 \nu_5) + \nu_7^6 (\nu_3^2 \nu_5 - \nu_5^2 - \nu_5^3 - \nu_5^4),$$

$$h = \sum_{j=0}^{5} \nu_7^j (1 - \nu_3 \nu_5) + \nu_7^6 (-\nu_3^2 + \nu_3 \nu_5^2 + \nu_3 \nu_5^3 + \nu_3 \nu_5^4),$$

respectively.

The next possibility is with top prime 7, weight partition $(2, 2, 2, 2, 2, 3, 3)$ and total weight 16. Suppose h is a minimal vanishing sorou with this data. Then h is as in (11) and after rotation, we may assume $1 \prec f_0$ and for a set $J = \{j(1), j(2)\}$ equal to one of $\{5, 6\}$, $\{4, 6\}$ or $\{3, 6\}$, we have

$$w(f_j) = \begin{cases} 3, & j \in J, \\ 2, & j \notin J. \end{cases}$$

We have $f_0 = 1 + b$ with b a root of unity not equal to -1 and having order dividing 30. As argued above, if $j \notin J$, then $f_j = f_0$. For $j \in J$, the type of $f_0 - f_j$ is either $R_2 \oplus R_3$ or R_5. By the observation made above (10), they cannot for both $j \in J$ be of type $R_2 \oplus R_3$.

If both types $R_2 \oplus R_3$ and R_5 occur, then without loss of generality, the type of $f_0 - f_{j(1)}$ is $R_2 \oplus R_3$ and the type of $f_0 - f_{j(2)}$ is R_5. After rotation, we must have $f_0 = 1 + \nu_5^y$ for some $y \in \{1, 2\}$. $f_{j(2)} = -\nu_5^{3-y} - \nu_5^3 - \nu_5^4$ and $f_{j(1)}$ equal to either $1 + \nu_5^y(-\nu_3 - \nu_3^2)$ or $\nu_5^y - \nu_3 - \nu_3^2$. Thus, we get the two minimal vanishing sorou

$$h = \left(\sum_{j \in J^c} \nu_7^j (1 + \nu_5^y) \right) + \nu_7^{j(1)} \left(1 + \nu_5^y(-\nu_3 - \nu_3^2) \right) + \nu_7^{j(2)} (-\nu_5^{3-y} - \nu_5^3 - \nu_5^4),$$

$$h = \left(\sum_{j \in J^c} \nu_7^j (1 + \nu_5^y) \right) + \nu_7^{j(1)} (\nu_5^y - \nu_3 - \nu_3^2) + \nu_7^{j(2)} (-\nu_5^{3-y} - \nu_5^3 - \nu_5^4),$$

each of type

$$(R_7 : 1 + \nu_5^y : R_2 \oplus R_3, R_5),$$

height 1 and with parities $(11, 5)$.

The case when the type of $f_0 - f_j$ is R_5 for both $j \in J$ is treated similarly. We get minimal vanishing sorou of type $(R_7 : 1 + \nu_5^y : 2R_5)$, all with height 1, order $70 = 2 \cdot 5 \cdot 7$ and parities $(10, 6)$. These cases were also treated in Theorem 3.2.

Regarding heights, if a minimal vanishing sorou h has type $(R_p : f_0 : T_1, \ldots, T_n)$, then, writing h as in (2), we see that the height of h is the maximum of the heights of the f_j for $0 \leq j \leq p-1$. Thus, the height of h is no greater than the maximum of the heights of f_0 and of the partitions $f_0 - f_j$ (when $f_j \neq f_0$) whose types are the subsidiary types $T_1, \ldots, T_n$. In particular, h can have height strictly greater than 1 only if either f_0 has height > 1 or some vanishing sorou of type T_i has height > 1, for some i. Examining Table 2, we see that for weight ≤ 15, all the f_0 have height 1 and all the subsidiary types are of minimal vanishing sorou. Thus, by induction, all minimal vanishing sorou of weight ≤ 15 have height 1.

The first example of a minimal vanishing sorou with non-minimal vanishing subsidiary types is the case of those of type $(R_7 : 1 + \nu_5^y : R_2 \oplus R_3, R_5)$, of weight 16. However, the analysis of these conducted above shows that they all have height 1.   ∎

4. Classification of vanishing sorou of weight > 16

To classify sorou of a large weight, computer assistance becomes indispensable. We designed an algorithm and coded it in python. It is essentially an exhaustive search, which we are able to do because results from previous sections imply that the search space is finite. The code is available in Appendix B as well as from the website `https://github.com/lchristie/Sums-of-Roots-of-Unity`. The main part of the algorithm is contained in `TypeGen.py` and described in Algorithm 4.1.

Algorithm 4.1. Algorithm 1 below uses the existing classification of minimal vanishing sorou up to weight $\leq k$ to find all minimal vanishing sorou of weight $k + 1$. We use the classification obtained in Section 3 to start if off. (We could have started this algorithm using list [PR98] of minimal vanishing sorou up to weight 12, due to Poonen and Rubinstein. However the derivation presented in Section 3 of minimal vanishing sorou up to weight 16 was important in order to develop understanding of and to present the expanded notion of type.)

We now describe how the algorithm finds all minimal vanishing types of weight $k + 1$. First note that the only possible top primes for a vanishing sorou of weight $k+1$ are $\leq k+1$. For each of these primes p we build the possible partitions of $k + 1$ into p summands. For each of these partitions we generate all possible f_0's (cf. Definition 2.4). Note that f_0 will be a sorou of relative order dividing the product of primes $< p$; this observation together with the bound $k + 1$ on the weight makes this a finite list.

Then for each of these f_0 we first prepare all possible lists of subtypes with weights given by the partition (this uses the already constructed type list of weight $< k + 1$). This is done carefully to eliminate redundancy and repetition. For each list we then build a new type of weight $k + 1$, generate a single sorou of this type and use this to check whether it is minimal vanishing with Proposition 2.3. To check that a sorou is vanishing we use symbolic manipulation with algebraic numbers in sympy.

It is now straightforward to verify that this algorithm generates all minimal vanishing types of weight $k + 1$.

4.1. **Heights and equisigned sum of roots of unity.** For use in [DK20], we will need to consider *equisigned* minimal vanishing sorou, namely, those whose parities $(n_{\text{odd}}, n_{\text{even}})$ satisfy $n_{\text{odd}} = n_{\text{even}}$. Given the recursive algorithm to describe types, it will be necessary to keep track of the possible parties for all types of sorou. To find all parities of a given type we again use computers. Our algorithm is coded in python. The code is available in Appendix B and from the website `https://github.com/lchristie/Sums-of-Roots-of-Unity`. The algorithm is contained in `sorouGenerator.py` and described in Algorithm 4.2.

Algorithm 4.2. To find all the parities we need to be able to find all sorou (up to rotation) of a given minimal vanishing type. We do this recursively per Algorithm 2 below, using the procedure *GenSorou*. For a given type $(R_p : f_0 : T_1, \ldots, T_n)$ we find all sorou of types T_i. We take combinations of one sorou from each T_i, and for each of these combinations

Algorithm 1 Generating all minimal vanishing types

1: **procedure** GENNEXTTYPES($previousTypes$)
2:     $Output \leftarrow \varnothing, previousWeight \leftarrow \max\{w(T) : T \in previousTypes\}$
3:     $w_0 \leftarrow previousWeight + 1$ ▷ Current weight to generate
4:     $primesToCheck \leftarrow \{p \in P : p \leq w_0\}$
5:     **for** p in $primesToCheck$ **do**
6:         $allParitions \leftarrow \{(x_0, x_1, \ldots, x_{p-1}) \in \mathbb{Z}_+^p : \sum x_i = w_0, 0 < x_i \leq x_j \text{ if } i \leq j\}$
7:         **for** x in $allParitions$ **do**
8:             **if** $x_0 = \cdots = x_{p-1} = 1$ **then return** R_p
9:             **else**
10:                 $F \leftarrow \{f \in S : w(f) = x_0, relOrder(f) | \prod_{q \in P}^{p-1} q\}$
11:                 **for** f_0 in F **do**
12:                     $X \leftarrow \mathbf{T}_{x_0+x_0} \times \cdots \times \mathbf{T}_{x_{p-1}+x_0}$ ▷ Possible subsidiary types
13:                     Filter X, based on the conditions below.
14:                     **for** $\mathbf{x}$ in X **do**
15:                         Generate one sorou h of type $T = (R_p : f_0 : x_0, \ldots, x_{p-1})$.
16:                         **if** h is minimal vanishing **then**   ▷ By Proposition 2.3
17:                             $Output \leftarrow Output \cup \{T\}$
18:     **return** $Output$

---

we subtract all suitable rotations of each sorou from the base sorou $f_{base} = \sum_{i=0}^{p-1} \nu_p^i f_0$ (i.e., rotations by ν_q^j where q is the relative order of the sorou, f_0, and ν_p concatenated as a formal sorou and $j \in \{0, 1, \ldots, q-1\}$, which cancel all terms of $f_0\nu_p^i$).

However, $T_1, \ldots, T_n$ are not necessarily minimal vanishing (cf. Remark 2.5 (f)). We thus also need to generate every sorou of a given *non-minimal* vanishing type $S_1 \oplus \cdots \oplus S_k$. But it must contain at least one term from f_0 in each minimal vanishing component, which makes the search space finite. We split f_0 into k subsorou and match sorou from each S_i to these subsorou. These sums of the sorou are then the possible sorou of type $S_1 \oplus \cdots \oplus S_k$ containing f_0. This is done via the procedure *GenNonMinSorou* below.

This means that the procedures *GenSorou* and *GenNonMinSorou* recursively call each other. Note however that this is guaranteed to terminate as each call forces a strict reduction in weight. Moreover, types with non-minimal subtypes only occur at weight 16 and higher, so for all of the computations done so far *GenNonMinSorou* never calls itself.

Once all the sorou of a given type have been constructed, it is easy to check their parities and heights.

4.2. **Results.** Appendix A contains a list of all minimal vanishing types of length ≤ 21 produced by this algorithm, as well as their heights and parities. In particular, we obtain the following.

Theorem 4.3.

(a) *All minimal vanishing sorou having weight less than 21 are of height* 1.
(b) *There are minimal vanishing sorou of weight 21 and height strictly greater than 1; they all have height* 2.

Algorithm 2 Generating All Possible Sorou of a Type

1: **procedure** GENSOROU($T = (R_p : f_0 : T_1, \ldots, T_n)$)
2:     $S_0 \leftarrow \{0\}, Output \leftarrow \varnothing$
3:     **for** i in $\{1, \ldots, n\}$ **do**
4:         $S_i \leftarrow GenNonMinSorou(T_i, f_0)$ ▷ As T_i may not be minimal
5:     $\Pi \leftarrow$ All Permutations of $(1, \ldots, n, 0, \ldots, 0)$ ▷ With $p - n > 0$ zeros
6:     **for** π in Π **do**
7:         $\Sigma_\pi \leftarrow S_{\pi(1)} \times \cdots \times S_{\pi(p)}$
8:         $f_{base} \leftarrow \sum_{i=0}^{p-1} f_0 \nu_p^i$
9:         **for** $(h_1, \ldots, h_p)$ in Σ_π **do**
10:             **for** i in $\{1, \ldots p\}$ **do**
11:                 $R_i \leftarrow \{\nu_a^b : f_0 \preceq \nu_a^b h_i\}$
12:             **for** $(\nu_{a_1}^{b_1}, \ldots, \nu_{a_p}^{b_p})$ in $R_1 \times \cdots \times R_p$ **do**
13:                 $g \leftarrow f_{base} - \sum_{i=1}^p \nu_p^i \nu_{a_i}^{b_i} h_i$ ▷ With subtracted terms removed
14:                 $Output \leftarrow Output \cup \{g\}$.
15:     **return** $Output$
16: **procedure** GENNONMINSOROU($T = T_1 \oplus \cdots \oplus T_m$, f)
17:     **if** $m = 1$ **then return** $GenSorou(T)$
18:     **else**
19:         $Output \leftarrow \varnothing$
20:         **for** i in $\{1, \ldots, m\}$ **do**
21:             $S_i \leftarrow GenSorou(T_i)$
22:         $\mathcal{P} \leftarrow \{(h_1, \ldots, h_m) : f_0 = \sum_{i=1}^m h_i, w(h_i) > 0\}$ ▷ "Partitions" of f_0
23:         $\Pi \leftarrow$ All permutations of $\{1, \ldots, m\}$
24:         $X \leftarrow \{(h, \pi, s) \in \mathcal{P} \times \Pi \times (S_1 \times \cdots \times S_m) : \forall i \exists \nu_{i \ni} h_{\pi(i)} \preceq \nu_i s_i\}$
25:         **for** $x \in X$ **do**
26:             $R_x \leftarrow \{(\nu_{a_1}^{b_1}, \ldots, \nu_{a_m}^{b_m}) : h_{\pi(i)} \preceq \nu_{a_i}^{b_i} s_i\}$
27:             **for** $(\nu_{a_1}^{b_1}, \ldots, \nu_{a_m}^{b_m})$ in R_x **do**
28:                 $g \leftarrow \sum_{i=1}^m \nu_{a_i}^{b_i} s_i$ ▷ So that $f_0 \preceq g$
29:                 $Output \leftarrow Output \cup \{g\}$
30:         **return** $Output$

Minimal vanishing sorou of arbitrarily large height have been constructed by Steinberger [Ste08], and their existence can be deduced from Schur's 1931 result (see [Leh36]) that cyclotomic polynomials Φ_n have unbounded coefficients. We point out that

$$\Phi_{105}(x) = x^{48} + x^{47} + x^{46} - x^{43} - x^{42} - 2x^{41} - x^{40} - x^{39} + x^{36} + x^{35} + x^{34} + x^{33} + x^{32} + x^{31} - x^{28}$$
$$- x^{26} - x^{24} - x^{22} - x^{20} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} - x^9 - x^8 - 2x^7 - x^6 - x^5 + x^2 + x + 1$$

is the first cyclotomic polynomial with a coefficient different from ± 1. It corresponds to a sorou of height 2 and weight 35.

References

[AS12] I. Aliev and C. Smyth, *Solving algebraic equations in roots of unity*, Forum Math. **24** (2012), no. 3, 641–665.

[CW11] T.-Y. Chien and S. Waldron, *A classification of the harmonic frames up to unitary equivalence*, Appl. Comput. Harmon. Anal. **30** (2011), no. 3, 307–318.

[CJ76] J. H. Conway and A. J. Jones, *Trigonometric Diophantine equations (On vanishing sums of roots of unity)*, Acta Arith. **30** (1976), no. 3, 229–240.

[DZ00] R. Dvornicich and U. Zannier, *On sums of roots of unity*, Monatsh. Math. **129** (2000), no. 2, 97–108.

[DK20] K. Dykema and I. Klep (2020), in preparation.

[FG14] R. M. Fonseca and W. Grimus, *Classification of lepton mixing matrices from finite residual symmetries*, Journal of High Energy Physics **2014** (2014), no. 9, 33 pages.

[JMS14] V. F. R. Jones, S. Morrison, and N. Snyder, *The classification of subfactors of index at most 5*, Bull. Amer. Math. Soc. (N.S.) **51** (2014), no. 2, 277–327.

[LL00] T. Y. Lam and K. H. Leung, *On vanishing sums of roots of unity*, J. Algebra **224** (2000), no. 1, 91–109.

[Leh36] E. Lehmer, *On the magnitude of the coefficients of the cyclotomic polynomial*, Bull. Amer. Math. Soc. **42** (1936), no. 6, 389–392.

[Len79] H. W. Lenstra, Jr., *Vanishing sums of roots of unity*, Proceedings, Bicentennial Congress Wiskundig Genootschap (Vrije Univ., Amsterdam, 1978), Part II, 1979, pp. 249–268.

[Man65] H. B. Mann, *On linear relations between roots of unity*, Mathematika **12** (1965), 107–117.

[PR98] B. Poonen and M. Rubinstein, *The number of intersection points made by the diagonals of a regular polygon*, SIAM J. Discrete Math. **11** (1998), 135–156.

[Sch64] I. J. Schoenberg, *A note on the cyclotomic polynomial*, Mathematika **11** (1964), 131–136.

[Ste08] J. P. Steinberger, *Minimal vanishing sums of roots of unity with large coefficients*, Proc. Lond. Math. Soc. (3) **97** (2008), no. 3, 689–717.

[Was97] L. C. Washington, *Introduction to cyclotomic fields*, Second, Graduate Texts in Mathematics, vol. 83, Springer-Verlag, New York, 1997.

Appendix A. List of all minimal vanishing types of length ≤ 21

Table 2: Minimal vanishing sums of roots of unity and their possible parities of orders.

Weight	Type	Possible heights	Possible parities
2	R_2	1	$(1,1)$
3	R_3	1	$(3,0)$
5	R_5	1	$(5,0)$
6	$(R_5 : R_3)$	1	$(4,2)$
7	$(R_5 : 2R_3)$	1	$(4,3)$
7	R_7	1	$(7,0)$
8	$(R_5 : 3R_3)$	1	$(6,2)$
8	$(R_7 : R_3)$	1	$(6,2)$
9	$(R_5 : 4R_3)$	1	$(8,1)$
9	$(R_7 : 2R_3)$	1	$(5,4)$
10	$(R_7 : R_5)$	1	$(6,4)$
10	$(R_7 : 3R_3)$	1	$(6,4)$
11	$(R_7 : (R_5 : R_3))$	1	$(8,3), (10,1)$
11	$(R_7 : R_5, R_3)$	1	$(6,5)$
11	$(R_7 : 4R_3)$	1	$(8,3)$
11	R_{11}	1	$(11,0)$
12	$(R_7 : (R_5 : 2R_3))$	1	$(9,3), (10,2)$
12	$(R_7 : R_3, (R_5 : R_3))$	1	$(7,5), (9,3)$
12	$(R_7 : 2R_3, R_5)$	1	$(8,4)$
12	$(R_7 : 5R_3)$	1	$(10,2)$
12	$(R_{11} : R_3)$	1	$(10,2)$
13	$(R_7 : (R_5 : 3R_3))$	1	$(8,5), (12,1)$
13	$(R_7 : (R_5 : 2R_3), R_3)$	1	$(8,5), (9,4)$
13	$(R_7 : 2R_5)$	1	$(8,5)$
13	$(R_7 : (R_5 : R_3), 2R_3)$	1	$(7,6), (8,5)$
13	$(R_7 : R_5, 3R_3)$	1	$(10,3)$
13	$(R_7 : 6R_3)$	1	$(12,1)$
13	$(R_{11} : 2R_3)$	1	$(9,4)$
13	R_{13}	1	$(13,0)$
14	$(R_7 : (R_5 : 4R_3))$	1	$(7,7), (14,0)$
14	$(R_7 : (R_5 : 3R_3), R_3)$	1	$(11,3), (7,7)$
14	$(R_7 : (R_5 : R_3), R_5)$	1	$(9,5), (7,7)$
14	$(R_7 : (R_5 : 2R_3), 2R_3)$	1	$(8,6), (7,7)$
14	$(R_7 : 2R_5, R_3)$	1	$(10,4)$
14	$(R_7 : (R_5 : R_3), 3R_3)$	1	$(9,5), (7,7)$
14	$(R_7 : R_5, 4R_3)$	1	$(12,2)$
14	$(R_{11} : R_5)$	1	$(10,4)$
14	$(R_{11} : 3R_3)$	1	$(8,6)$
14	$(R_{13} : R_3)$	1	$(12,2)$
15	$(R_7 : (R_5 : 4R_3), R_3)$	1	$(9,6), (13,2)$
15	$(R_7 : (R_5 : 2R_3), R_5)$	1	$(9,6), (8,7)$
15	$(R_7 : 2(R_5 : R_3))$	1	$(11,4), (9,6), (13,2)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
15	$(R_7 : (R_5 : 3R_3), 2R_3)$	1	$(10,5), (9,6)$
15	$(R_7 : (R_5 : R_3), R_5, R_3)$	1	$(9,6), (8,7)$
15	$(R_7 : (R_5 : 2R_3), 3R_3)$	1	$(9,6), (8,7)$
15	$(R_7 : 2R_5, 2R_3)$	1	$(12,3)$
15	$(R_7 : (R_5 : R_3), 4R_3)$	1	$(11,4), (9,6)$
15	$(R_7 : R_5, 5R_3)$	1	$(14,1)$
15	$(R_7 : 1 + \nu_5^1 : R_5)$	1	$(12,3)$
15	$(R_7 : 1 + \nu_5^2 : R_5)$	1	$(12,3)$
15	$(R_{11} : (R_5 : R_3))$	1	$(14,1), (12,3)$
15	$(R_{11} : R_5, R_3)$	1	$(9,6)$
15	$(R_{11} : 4R_3)$	1	$(8,7)$
15	$(R_{13} : 2R_3)$	1	$(11,4)$
16	$(R_7 : (R_5 : 3R_3), R_5)$	1	$(11,5), (9,7)$
16	$(R_7 : (R_5 : 2R_3), (R_5 : R_3))$	1	$(12,4), (10,6), (11,5), (13,3)$
16	$(R_7 : (R_5 : 4R_3), 2R_3)$	1	$(12,4), (11,5)$
16	$(R_7 : (R_5 : 2R_3), R_5, R_3)$	1	$(8,8), (9,7)$
16	$(R_7 : 2(R_5 : R_3), R_3)$	1	$(8,8), (10,6), (12,4)$
16	$(R_7 : 3R_5)$	1	$(12,4)$
16	$(R_7 : (R_5 : 3R_3), 3R_3)$	1	$(9,7), (11,5)$
16	$(R_7 : (R_5 : R_3), R_5, 2R_3)$	1	$(9,7), (11,5)$
16	$(R_7 : (R_5 : 2R_3), 4R_3)$	1	$(10,6), (11,5)$
16	$(R_7 : 2R_5, 3R_3)$	1	$(14,2)$
16	$(R_7 : (R_5 : R_3), 5R_3)$	1	$(11,5), (13,3)$
16	$(R_7 : 1 + \nu_3^1 : (R_5 : R_3))$	1	$(16,0)$
16	$(R_7 : 1 + \nu_{30}^1 : (R_5 : R_3))$	1	$(9,7)$
16	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3))$	1	$(14,2)$
16	$(R_7 : 1 + \nu_{30}^7 : (R_5 : R_3))$	1	$(9,7)$
16	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3))$	1	$(9,7)$
16	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3))$	1	$(14,2)$
16	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3))$	1	$(9,7)$
16	$(R_7 : 1 + \nu_5^1 : 2R_5)$	1	$(10,6)$
16	$(R_7 : 1 + \nu_5^2 : 2R_5)$	1	$(10,6)$
16	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_2), R_5)$	1	$(11,5)$
16	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_2), R_5)$	1	$(11,5)$
16	$(R_{11} : R_7)$	1	$(10,6)$
16	$(R_{11} : (R_5 : 2R_3))$	1	$(14,2), (13,3)$
16	$(R_{11} : (R_5 : R_3), R_3)$	1	$(13,3), (11,5)$
16	$(R_{11} : R_5, 2R_3)$	1	$(8,8)$
16	$(R_{11} : 5R_3)$	1	$(10,6)$
16	$(R_{13} : R_5)$	1	$(12,4)$
16	$(R_{13} : 3R_3)$	1	$(10,6)$
17	$(R_7 : (R_5 : 4R_3), R_5)$	1	$(13,4), (11,6)$
17	$(R_7 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(9,8), (13,4), (11,6), (15,2)$
17	$(R_7 : 2(R_5 : 2R_3))$	1	$(12,5), (13,4), (11,6)$
17	$(R_7 : (R_5 : 3R_3), R_5, R_3)$	1	$(10,7), (11,6)$
17	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), R_3)$	1	$(12,5), (10,7), (9,8), (11,6)$
17	$(R_7 : (R_5 : R_3), 2R_5)$	1	$(9,8), (11,6)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
17	$(R_7 : (R_5 : 4R_3), 3R_3)$	1	$(13,4), (11,6)$
17	$(R_7 : (R_5 : 2R_3), R_5, 2R_3)$	1	$(10,7), (11,6)$
17	$(R_7 : 2(R_5 : R_3), 2R_3)$	1	$(10,7), (9,8), (11,6)$
17	$(R_7 : 3R_5, R_3)$	1	$(14,3)$
17	$(R_7 : (R_5 : 3R_3), 4R_3)$	1	$(13,4), (9,8)$
17	$(R_7 : (R_5 : R_3), R_5, 3R_3)$	1	$(13,4), (11,6)$
17	$(R_7 : (R_5 : 2R_3), 5R_3)$	1	$(12,5), (13,4)$
17	$(R_7 : 2R_5, 4R_3)$	1	$(16,1)$
17	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3))$	1	$(9,8)$
17	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3))$	1	$(15,2)$
17	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3))$	1	$(16,1), (15,2)$
17	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3))$	1	$(9,8)$
17	$(R_7 : 1 + \nu_3^1 : (R_5 : 2R_3))$	1	$(15,2)$
17	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3))$	1	$(9,8)$
17	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3))$	1	$(16,1), (15,2)$
17	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3))$	1	$(9,8)$
17	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3))$	1	$(15,2)$
17	$(R_7 : 1 + \nu_{30}^1 : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(9,8), (11,6)$
17	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), R_5)$	1	$(12,5)$
17	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,4)$
17	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), R_5)$	1	$(10,7)$
17	$(R_7 : 1 + \nu_{30}^7 : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(9,8), (11,6)$
17	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(9,8), (11,6)$
17	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), R_5)$	1	$(12,5)$
17	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,4)$
17	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), R_5)$	1	$(10,7)$
17	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3), (R_3 \oplus R_2))$	1	$(9,8), (11,6)$
17	$(R_7 : 1 + \nu_5^1 : 3R_5)$	1	$(9,8)$
17	$(R_7 : 1 + \nu_5^2 : 3R_5)$	1	$(9,8)$
17	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_2), 2R_5)$	1	$(9,8)$
17	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_2), R_5)$	1	$(10,7)$
17	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_2), 2R_5)$	1	$(9,8)$
17	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_2), R_5)$	1	$(10,7)$
17	$(R_{11} : (R_5 : 3R_3))$	1	$(12,5), (16,1)$
17	$(R_{11} : (R_7 : R_3))$	1	$(12,5), (16,1)$
17	$(R_{11} : (R_5 : 2R_3), R_3)$	1	$(12,5), (13,4)$
17	$(R_{11} : R_7, R_3)$	1	$(9,8)$
17	$(R_{11} : 2R_5)$	1	$(9,8)$
17	$(R_{11} : (R_5 : R_3), 2R_3)$	1	$(10,7), (12,5)$
17	$(R_{11} : R_5, 3R_3)$	1	$(10,7)$
17	$(R_{11} : 6R_3)$	1	$(12,5)$
17	$(R_{13} : (R_5 : R_3))$	1	$(14,3), (16,1)$
17	$(R_{13} : R_5, R_3)$	1	$(11,6)$
17	$(R_{13} : 4R_3)$	1	$(9,8)$
17	R_{17}	1	$(17,0)$
18	$(R_7 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15,3), (10,8), (17,1)$
18	$(R_7 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(15,3), (10,8), (11,7), (14,4)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
18	$(R_7 : (R_5 : 4R_3), R_5, R_3)$	1	$(12,6), (13,5)$
18	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), R_3)$	1	$(10,8), (12,6), (14,4)$
18	$(R_7 : 2(R_5 : 2R_3), R_3)$	1	$(11,7), (12,6), (10,8)$
18	$(R_7 : (R_5 : 2R_3), 2R_5)$	1	$(10,8), (11,7)$
18	$(R_7 : 2(R_5 : R_3), R_5)$	1	$(10,8), (12,6)$
18	$(R_7 : (R_5 : 3R_3), R_5, 2R_3)$	1	$(9,9), (13,5)$
18	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), 2R_3)$	1	$(11,7), (9,9), (10,8)$
18	$(R_7 : (R_5 : R_3), 2R_5, R_3)$	1	$(11,7), (13,5)$
18	$(R_7 : (R_5 : 4R_3), 4R_3)$	1	$(15,3), (10,8)$
18	$(R_7 : (R_5 : 2R_3), R_5, 3R_3)$	1	$(13,5), (12,6)$
18	$(R_7 : 2(R_5 : R_3), 3R_3)$	1	$(10,8), (12,6)$
18	$(R_7 : 3R_5, 2R_3)$	1	$(16,2)$
18	$(R_7 : (R_5 : 3R_3), 5R_3)$	1	$(15,3), (11,7)$
18	$(R_7 : (R_5 : R_3), R_5, 4R_3)$	1	$(15,3), (13,5)$
18	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3))$	1	$(11,7)$
18	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3))$	1	$(14,4), (18,0)$
18	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3))$	1	$(11,7)$
18	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_3^1 : (R_5 : 3R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3))$	1	$(11,7)$
18	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3))$	1	$(14,4), (18,0)$
18	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3))$	1	$(11,7)$
18	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11,7), (10,8)$
18	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), R_5)$	1	$(14,4), (13,5)$
18	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15,3), (14,4)$
18	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(10,8), (11,7)$
18	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(10,8), (11,7)$
18	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), R_5)$	1	$(14,4), (13,5)$
18	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15,3), (14,4)$
18	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11,7), (10,8)$
18	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_3^1 : 2(R_5 : R_3))$	1	$(18,0)$
18	$(R_7 : 1 + \nu_{30}^1 : 2(R_5 : R_3))$	1	$(11,7), (9,9)$
18	$(R_7 : 1 + \nu_{30}^1 : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(10,8)$
18	$(R_7 : 1 + \nu_5^1 : 2(R_5 : R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(12,6)$
18	$(R_7 : 1 + \nu_{30}^7 : 2(R_5 : R_3))$	1	$(11,7), (9,9)$
18	$(R_7 : 1 + \nu_{30}^7 : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(10,8)$
18	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : R_3))$	1	$(11,7), (9,9)$
18	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(10,8)$
18	$(R_7 : 1 + \nu_5^2 : 2(R_5 : R_3))$	1	$(14,4)$
18	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(12,6)$
18	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : R_3))$	1	$(11,7), (9,9)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
18	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), (R_5 : R_3))$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{30}^{1} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11, 7), (13, 5), (10, 8)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 7)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(12, 6)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(9, 9)$
18	$(R_7 : 1 + \nu_{30}^{7} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11, 7), (13, 5), (10, 8)$
18	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11, 7), (13, 5), (10, 8)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : R_3), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 7)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(12, 6)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(9, 9)$
18	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11, 7), (13, 5), (10, 8)$
18	$(R_7 : 1 + \nu_{5}^{1} : 4R_5)$	1	$(12, 6)$
18	$(R_7 : 1 + \nu_{5}^{2} : 4R_5)$	1	$(12, 6)$
18	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_2), 3R_5)$	1	$(11, 7)$
18	$(R_7 : 1 + \nu_{5}^{1} : 2(R_3 \oplus R_2), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{1} : 3(R_3 \oplus R_2), R_5)$	1	$(9, 9)$
18	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_2), 3R_5)$	1	$(11, 7)$
18	$(R_7 : 1 + \nu_{5}^{2} : 2(R_3 \oplus R_2), 2R_5)$	1	$(10, 8)$
18	$(R_7 : 1 + \nu_{5}^{2} : 3(R_3 \oplus R_2), R_5)$	1	$(9, 9)$
18	$(R_{11} : (R_5 : 4R_3))$	1	$(11, 7), (18, 0)$
18	$(R_{11} : (R_7 : 2R_3))$	1	$(15, 3), (14, 4)$
18	$(R_{11} : (R_5 : 3R_3), R_3)$	1	$(15, 3), (11, 7)$
18	$(R_{11} : (R_7 : R_3), R_3)$	1	$(15, 3), (11, 7)$
18	$(R_{11} : (R_5 : R_3), R_5)$	1	$(11, 7), (13, 5)$
18	$(R_{11} : (R_5 : 2R_3), 2R_3)$	1	$(11, 7), (12, 6)$
18	$(R_{11} : R_7, 2R_3)$	1	$(10, 8)$
18	$(R_{11} : 2R_5, R_3)$	1	$(10, 8)$
18	$(R_{11} : (R_5 : R_3), 3R_3)$	1	$(11, 7), (9, 9)$
18	$(R_{11} : R_5, 4R_3)$	1	$(12, 6)$
18	$(R_{11} : 7R_3)$	1	$(14, 4)$
18	$(R_{13} : R_7)$	1	$(12, 6)$
18	$(R_{13} : (R_5 : 2R_3))$	1	$(15, 3), (16, 2)$
18	$(R_{13} : (R_5 : R_3), R_3)$	1	$(15, 3), (13, 5)$
18	$(R_{13} : R_5, 2R_3)$	1	$(10, 8)$
18	$(R_{13} : 5R_3)$	1	$(10, 8)$
18	$(R_{17} : R_3)$	1	$(16, 2)$
19	$(R_7 : (R_5 : 4R_3), (R_5 : 2R_3))$	1	$(17, 2), (16, 3), (10, 9)$
19	$(R_7 : 2(R_5 : 3R_3))$	1	$(13, 6), (17, 2), (10, 9)$
19	$(R_7 : (R_5 : 4R_3), (R_5 : R_3), R_3)$	1	$(12, 7), (16, 3), (14, 5), (10, 9)$
19	$(R_7 : (R_5 : 3R_3), (R_5 : 2R_3), R_3)$	1	$(13, 6), (14, 5), (10, 9)$
19	$(R_7 : (R_5 : 3R_3), 2R_5)$	1	$(13, 6), (10, 9)$
19	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), R_5)$	1	$(11, 8), (12, 7), (10, 9)$
19	$(R_7 : 3(R_5 : R_3))$	1	$(12, 7), (16, 3), (10, 9), (14, 5)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
19	$(R_7 : (R_5 : 4R_3), R_5, 2R_3)$	1	$(15,4), (11,8)$
19	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), 2R_3)$	1	$(13,6), (12,7), (11,8), (10,9)$
19	$(R_7 : 2(R_5 : 2R_3), 2R_3)$	1	$(11,8), (10,9)$
19	$(R_7 : (R_5 : 2R_3), 2R_5, R_3)$	1	$(13,6), (12,7)$
19	$(R_7 : 2(R_5 : R_3), R_5, R_3)$	1	$(11,8), (12,7), (10,9)$
19	$(R_7 : 4R_5)$	1	$(16,3)$
19	$(R_7 : (R_5 : 3R_3), R_5, 3R_3)$	1	$(15,4), (11,8)$
19	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), 3R_3)$	1	$(11,8), (12,7), (10,9)$
19	$(R_7 : (R_5 : R_3), 2R_5, 2R_3)$	1	$(15,4), (13,6)$
19	$(R_7 : (R_5 : 4R_3), 5R_3)$	1	$(17,2), (10,9)$
19	$(R_7 : (R_5 : 2R_3), R_5, 4R_3)$	1	$(15,4), (14,5)$
19	$(R_7 : 2(R_5 : R_3), 4R_3)$	1	$(12,7), (14,5), (10,9)$
19	$(R_7 : 3R_5, 3R_3)$	1	$(18,1)$
19	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_3^1 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 4R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (10,9)$
19	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), R_5)$	1	$(12,7), (16,3)$
19	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (17,2)$
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (10,9)$
19	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (10,9)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), R_5)$	1	$(12,7), (16,3)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (17,2)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6), (10,9)$
19	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), (R_3 \oplus R_2))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_5 : R_3))$	1	$(11,8), (10,9)$
19	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(10,9)$
19	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_5 : R_3))$	1	$(13,6), (11,8)$
19	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3))$	1	$(15,4), (16,3)$
19	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(13,6), (14,5)$
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_5 : R_3))$	1	$(11,8), (10,9)$
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(10,9)$
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_5 : R_3))$	1	$(11,8), (13,6)$
19	$(R_7 : 1 + \nu_3^1 : (R_5 : 2R_3), (R_5 : R_3))$	1	$(17,2)$
19	$(R_7 : 1 + \nu_3^1 : (R_5 \oplus R_2), (R_5 : R_3))$	1	$(15,4)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_5 : R_3))$	1	$(11, 8), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_5 : R_3))$	1	$(13, 6), (11, 8)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), (R_5 : R_3))$	1	$(15, 4), (16, 3)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(13, 6), (14, 5)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_5 : R_3))$	1	$(11, 8), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_5 : R_3))$	1	$(13, 6), (11, 8)$
19	$(R_7 : 1 + \nu_{15}^{7} : (R_5 : 2R_3), (R_3 \oplus R_3))$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{30}^{1} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{15}^{2} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 2R_5)$	1	$(11, 8), (12, 7)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), (R_3 \oplus R_2), R_5)$	1	$(13, 6), (12, 7)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (14, 5)$
19	$(R_7 : 1 + \nu_{30}^{7} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 2R_5)$	1	$(11, 8), (12, 7)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), (R_3 \oplus R_2), R_5)$	1	$(13, 6), (12, 7)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (14, 5)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6), (12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{15}^{7} : (R_5 : 2R_3), 2(R_3 \oplus R_2))$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{30}^{1} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 8), (10, 9), (13, 6)$
19	$(R_7 : 1 + \nu_{30}^{1} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), R_5)$	1	$(12, 7)$
19	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), R_5)$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{5}^{1} : 2(R_3 \oplus R_3), R_5)$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{30}^{7} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 8), (10, 9), (13, 6)$
19	$(R_7 : 1 + \nu_{30}^{7} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 6), (11, 8), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), R_5)$	1	$(12, 7)$
19	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), (R_5 : R_3), R_5)$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{5}^{2} : 2(R_3 \oplus R_3), R_5)$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 6), (11, 8), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12, 7), (10, 9)$
19	$(R_7 : 1 + \nu_{30}^{1} : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(13, 6), (12, 7), (15, 4), (10, 9)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), 3R_5)$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(10, 9)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 8)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), 3R_5)$	1	$(13, 6)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12, 7)$
19	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11, 8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
19	$(R_7 : 1 + \nu_{30}^7 : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(13,6), (12,7), (15,4), (10,9)$
19	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(15,4), (12,7), (10,9), (13,6)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 3R_5)$	1	$(11,8)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(10,9)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(10,9)$
19	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11,8)$
19	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 3R_5)$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12,7)$
19	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11,8)$
19	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(15,4), (12,7), (10,9), (13,6)$
19	$(R_7 : 1 + \nu_5^1 : 5R_5)$	1	$(15,4)$
19	$(R_7 : 1 + \nu_5^2 : 5R_5)$	1	$(15,4)$
19	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_2), 4R_5)$	1	$(14,5)$
19	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_2), 3R_5)$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^1 : 3(R_3 \oplus R_2), 2R_5)$	1	$(12,7)$
19	$(R_7 : 1 + \nu_5^1 : 4(R_3 \oplus R_2), R_5)$	1	$(11,8)$
19	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_2), 4R_5)$	1	$(14,5)$
19	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_2), 3R_5)$	1	$(13,6)$
19	$(R_7 : 1 + \nu_5^2 : 3(R_3 \oplus R_2), 2R_5)$	1	$(12,7)$
19	$(R_7 : 1 + \nu_5^2 : 4(R_3 \oplus R_2), R_5)$	1	$(11,8)$
19	$(R_{11} : (R_7 : 3R_3))$	1	$(16,3), (14,5)$
19	$(R_{11} : (R_7 : R_5))$	1	$(16,3), (14,5)$
19	$(R_{11} : (R_5 : 4R_3), R_3)$	1	$(17,2), (10,9)$
19	$(R_{11} : (R_7 : 2R_3), R_3)$	1	$(13,6), (14,5)$
19	$(R_{11} : (R_5 : 2R_3), R_5)$	1	$(13,6), (12,7)$
19	$(R_{11} : R_7, R_5)$	1	$(10,9)$
19	$(R_{11} : 2(R_5 : R_3))$	1	$(13,6), (17,2), (15,4)$
19	$(R_{11} : (R_5 : 3R_3), 2R_3)$	1	$(14,5), (10,9)$
19	$(R_{11} : (R_7 : R_3), 2R_3)$	1	$(14,5), (10,9)$
19	$(R_{11} : (R_5 : R_3), R_5, R_3)$	1	$(12,7), (10,9)$
19	$(R_{11} : (R_5 : 2R_3), 3R_3)$	1	$(11,8), (10,9)$
19	$(R_{11} : R_7, 3R_3)$	1	$(12,7)$
19	$(R_{11} : 2R_5, 2R_3)$	1	$(12,7)$
19	$(R_{11} : (R_5 : R_3), 4R_3)$	1	$(11,8), (10,9)$
19	$(R_{11} : R_5, 5R_3)$	1	$(14,5)$
19	$(R_{11} : 8R_3)$	1	$(16,3)$
19	$(R_{13} : (R_5 : 3R_3))$	1	$(14,5), (18,1)$
19	$(R_{13} : (R_7 : R_3))$	1	$(14,5), (18,1)$
19	$(R_{13} : (R_5 : 2R_3), R_3)$	1	$(15,4), (14,5)$
19	$(R_{13} : R_7, R_3)$	1	$(11,8)$
19	$(R_{13} : 2R_5)$	1	$(11,8)$
19	$(R_{13} : (R_5 : R_3), 2R_3)$	1	$(12,7), (14,5)$
19	$(R_{13} : R_5, 3R_3)$	1	$(10,9)$
19	$(R_{13} : 6R_3)$	1	$(12,7)$
19	$(R_{17} : 2R_3)$	1	$(15,4)$
19	R_{19}	1	$(19,0)$
20	$(R_7 : (R_5 : 4R_3), (R_5 : 3R_3))$	1	$(15,5), (19,1), (12,8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_7 : (R_5 : 4R_3), (R_5 : 2R_3), R_3)$	1	$(16,4),(15,5),(11,9),(12,8)$
20	$(R_7 : 2(R_5 : 3R_3), R_3)$	1	$(16,4),(12,8)$
20	$(R_7 : (R_5 : 4R_3), 2R_5)$	1	$(15,5),(12,8)$
20	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), R_5)$	1	$(14,6),(12,8),(10,10)$
20	$(R_7 : 2(R_5 : 2R_3), R_5)$	1	$(11,9),(12,8),(10,10)$
20	$(R_7 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(16,4),(11,9),(14,6),(13,7),$
			$(15,5),(12,8)$
20	$(R_7 : (R_5 : 4R_3), (R_5 : R_3), 2R_3)$	1	$(13,7),(14,6),(15,5),(12,8)$
20	$(R_7 : (R_5 : 3R_3), (R_5 : 2R_3), 2R_3)$	1	$(13,7),(11,9),(12,8)$
20	$(R_7 : (R_5 : 3R_3), 2R_5, R_3)$	1	$(15,5),(11,9)$
20	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), R_5, R_3)$	1	$(11,9),(12,8),(10,10)$
20	$(R_7 : 3(R_5 : R_3), R_3)$	1	$(13,7),(11,9),(15,5)$
20	$(R_7 : (R_5 : R_3), 3R_5)$	1	$(13,7),(15,5)$
20	$(R_7 : (R_5 : 4R_3), R_5, 3R_3)$	1	$(17,3),(10,10)$
20	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), 3R_3)$	1	$(14,6),(10,10),(12,8)$
20	$(R_7 : 2(R_5 : 2R_3), 3R_3)$	1	$(10,10),(11,9),(12,8)$
20	$(R_7 : (R_5 : 2R_3), 2R_5, 2R_3)$	1	$(15,5),(14,6)$
20	$(R_7 : 2(R_5 : R_3), R_5, 2R_3)$	1	$(14,6),(10,10),(12,8)$
20	$(R_7 : 4R_5, R_3)$	1	$(18,2)$
20	$(R_7 : (R_5 : 3R_3), R_5, 4R_3)$	1	$(13,7),(17,3)$
20	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), 4R_3)$	1	$(13,7),(14,6),(11,9),(12,8)$
20	$(R_7 : (R_5 : R_3), 2R_5, 3R_3)$	1	$(17,3),(15,5)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(15,5),(12,8)$
20	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(15,5),(12,8)$
20	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(15,5),(12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(15,5),(12,8)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 4R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(13,7),(11,9)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(18,2),(14,6)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(16,4),(12,8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(13,7),(11,9)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^7 : ((R_5 : R_3) \oplus R_2), (R_5 : R_3))$	1	$(12,8),(10,10)$
20	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_3^1 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(16,4)$
20	$(R_7 : 1 + \nu_3^1 : ((R_5 : R_3) \oplus R_2), (R_5 : R_3))$	1	$(17,3)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_5 : R_3))$	1	$(13,7),(11,9)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_5 : R_3))$	1	$(18, 2), (14, 6)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(16, 4), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_5 : R_3))$	1	$(13, 7), (11, 9)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_{30}^{13} : ((R_5 : R_3) \oplus R_2), (R_5 : R_3))$	1	$(10, 10), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), (R_3 \oplus R_3))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_{30}^1 : 2(R_5 : 2R_3))$	1	$(11, 9), (10, 10)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(13, 7), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^2 : 2(R_5 : 2R_3))$	1	$(16, 4)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_5 : 2R_3))$	1	$(17, 3), (18, 2), (16, 4)$
20	$(R_7 : 1 + \nu_{30}^7 : 2(R_5 : 2R_3))$	1	$(11, 9), (10, 10)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(13, 7), (12, 8)$
20	$(R_7 : 1 + \nu_3^1 : 2(R_5 : 2R_3))$	1	$(16, 4)$
20	$(R_7 : 1 + \nu_3^1 : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : 2R_3))$	1	$(11, 9), (10, 10)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(13, 7), (12, 8)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_5 : 2R_3))$	1	$(16, 4), (17, 3), (18, 2)$
20	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : 2R_3))$	1	$(11, 9), (10, 10)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(13, 7), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^7 : 2(R_5 : 2R_3))$	1	$(16, 4)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 \oplus R_2), (R_5 : 2R_3))$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(11, 9), (15, 5), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), 2R_5)$	1	$(14, 6), (10, 10)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 9), (15, 5)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(16, 4), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(15, 5), (11, 9), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(11, 9), (15, 5), (12, 8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), 2R_5)$	1	$(14, 6), (10, 10)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 9), (15, 5)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(16, 4), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(11, 9), (15, 5), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), 2(R_3 \oplus R_2))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(10, 10), (13, 7), (11, 9), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 9), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 7), (10, 10), (15, 5), (12, 8)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(12, 8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3), R_5)$	1	$(13, 7), (14, 6)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(15, 5), (14, 6)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), R_5)$	1	$(11, 9), (12, 8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(13, 7), (12, 8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13, 7), (11, 9), (12, 8), (10, 10)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 9), (12, 8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,7), (15,5), (12,8), (10,10)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(10,10), (13,7), (11,9), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11,9), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,7), (10,10), (15,5), (12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_5 : R_3), R_5)$	1	$(13,7), (14,6)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(15,5), (14,6)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), R_5)$	1	$(11,9), (12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(13,7), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(10,10), (13,7), (11,9), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11,9), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,7), (10,10), (15,5), (12,8)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_3^1 : 3(R_5 : R_3))$	1	$(20,0)$
20	$(R_7 : 1 + \nu_{30}^1 : 3(R_5 : R_3))$	1	$(13,7), (11,9)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(12,8), (10,10)$
20	$(R_7 : 1 + \nu_{30}^1 : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^1 : 3(R_5 : R_3))$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_{30}^7 : 3(R_5 : R_3))$	1	$(13,7), (11,9)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(12,8), (10,10)$
20	$(R_7 : 1 + \nu_{30}^7 : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,9)$
20	$(R_7 : 1 + \nu_{30}^{11} : 3(R_5 : R_3))$	1	$(13,7), (11,9)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(10,10), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^2 : 3(R_5 : R_3))$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_{30}^{13} : 3(R_5 : R_3))$	1	$(13,7), (11,9)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), 2(R_5 : R_3))$	1	$(12,8), (10,10)$
20	$(R_7 : 1 + \nu_{30}^{13} : 2(R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,9)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(15,5), (11,9), (12,8), (14,6)$
20	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), 3R_5)$	1	$(11,9), (10,10)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_2), 2R_5)$	1	$(11,9), (10,10)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11,9), (12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(13,7), (12,8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(15,5), (12,8), (11,9), (14,6)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(15,5), (12,8), (11,9), (14,6)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), 3R_5)$	1	$(11,9), (10,10)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_2), 2R_5)$	1	$(11,9), (10,10)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11,9), (12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(13,7), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(15,5), (11,9), (12,8), (14,6)$
20	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3), 3(R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_{30}^1 : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10), (11,9), (13,7), (15,5),$ $(12,8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_7 : 1 + \nu_{30}^1 : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,9), (12,8), (14,6)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_5 : R_3), 2R_5)$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_5 : R_3), 2R_5)$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_3), 2R_5)$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(13,7)$
20	$(R_7 : 1 + \nu_{30}^7 : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10), (11,9), (13,7), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(14,6), (11,9), (12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10), (11,9), (13,7), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,9), (12,8), (14,6)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_5 : R_3), 2R_5)$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3), 2R_5)$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), 2R_5)$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(13,7)$
20	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(10,10), (11,9), (13,7), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,9), (12,8), (14,6)$
20	$(R_7 : 1 + \nu_{30}^1 : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(11,9), (14,6), (17,3), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 4R_5)$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), (R_3 \oplus R_2), 3R_5)$	1	$(13,7)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 3(R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 4R_5)$	1	$(16,4)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_3 \oplus R_2), 3R_5)$	1	$(15,5)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 3(R_3 \oplus R_2), R_5)$	1	$(13,7)$
20	$(R_7 : 1 + \nu_{30}^7 : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(11,9), (14,6), (17,3), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(11,9), (14,6), (17,3), (15,5),$ $(12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 4R_5)$	1	$(14,6)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), (R_3 \oplus R_2), 3R_5)$	1	$(13,7)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(12,8)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 3(R_3 \oplus R_2), R_5)$	1	$(11,9)$
20	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(10,10)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 4R_5)$	1	$(16,4)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_3 \oplus R_2), 3R_5)$	1	$(15,5)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 3(R_3 \oplus R_2), R_5)$	1	$(13, 7)$
20	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3), 4(R_3 \oplus R_2))$	1	$(11, 9), (14, 6), (17, 3), (15, 5),$ $(12, 8)$
20	$(R_7 : 1 + \nu_5^1 : 6R_5)$	1	$(18, 2)$
20	$(R_7 : 1 + \nu_5^2 : 6R_5)$	1	$(18, 2)$
20	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_2), 5R_5)$	1	$(17, 3)$
20	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_2), 4R_5)$	1	$(16, 4)$
20	$(R_7 : 1 + \nu_5^1 : 3(R_3 \oplus R_2), 3R_5)$	1	$(15, 5)$
20	$(R_7 : 1 + \nu_5^1 : 4(R_3 \oplus R_2), 2R_5)$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_5^1 : 5(R_3 \oplus R_2), R_5)$	1	$(13, 7)$
20	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_2), 5R_5)$	1	$(17, 3)$
20	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_2), 4R_5)$	1	$(16, 4)$
20	$(R_7 : 1 + \nu_5^2 : 3(R_3 \oplus R_2), 3R_5)$	1	$(15, 5)$
20	$(R_7 : 1 + \nu_5^2 : 4(R_3 \oplus R_2), 2R_5)$	1	$(14, 6)$
20	$(R_7 : 1 + \nu_5^2 : 5(R_3 \oplus R_2), R_5)$	1	$(13, 7)$
20	$(R_{11} : (R_7 : 4R_3))$	1	$(13, 7), (18, 2)$
20	$(R_{11} : (R_7 : (R_5 : R_3)))$	1	$(20, 0), (13, 7), (11, 9), (18, 2)$
20	$(R_{11} : (R_7 : R_5, R_3))$	1	$(15, 5), (16, 4)$
20	$(R_{11} : (R_7 : 3R_3), R_3)$	1	$(13, 7), (15, 5)$
20	$(R_{11} : (R_7 : R_5), R_3)$	1	$(13, 7), (15, 5)$
20	$(R_{11} : (R_5 : 3R_3), R_5)$	1	$(11, 9), (15, 5)$
20	$(R_{11} : (R_7 : R_3), R_5)$	1	$(11, 9), (15, 5)$
20	$(R_{11} : (R_5 : 2R_3), (R_5 : R_3))$	1	$(16, 4), (15, 5), (17, 3), (14, 6)$
20	$(R_{11} : R_7, (R_5 : R_3))$	1	$(13, 7), (11, 9)$
20	$(R_{11} : (R_5 : 4R_3), 2R_3)$	1	$(11, 9), (16, 4)$
20	$(R_{11} : (R_7 : 2R_3), 2R_3)$	1	$(13, 7), (12, 8)$
20	$(R_{11} : (R_5 : 2R_3), R_5, R_3)$	1	$(11, 9), (12, 8)$
20	$(R_{11} : R_7, R_5, R_3)$	1	$(12, 8)$
20	$(R_{11} : 2(R_5 : R_3), R_3)$	1	$(14, 6), (16, 4), (12, 8)$
20	$(R_{11} : 3R_5)$	1	$(12, 8)$
20	$(R_{11} : (R_5 : 3R_3), 3R_3)$	1	$(13, 7), (11, 9)$
20	$(R_{11} : (R_7 : R_3), 3R_3)$	1	$(13, 7), (11, 9)$
20	$(R_{11} : (R_5 : R_3), R_5, 2R_3)$	1	$(11, 9)$
20	$(R_{11} : (R_5 : 2R_3), 4R_3)$	1	$(11, 9), (10, 10)$
20	$(R_{11} : R_7, 4R_3)$	1	$(14, 6)$
20	$(R_{11} : 2R_5, 3R_3)$	1	$(14, 6)$
20	$(R_{11} : (R_5 : R_3), 5R_3)$	1	$(13, 7), (11, 9)$
20	$(R_{11} : R_5, 6R_3)$	1	$(16, 4)$
20	$(R_{11} : 9R_3)$	1	$(18, 2)$
20	$(R_{13} : (R_5 : 4R_3))$	1	$(13, 7), (20, 0)$
20	$(R_{13} : (R_7 : 2R_3))$	1	$(17, 3), (16, 4)$
20	$(R_{13} : (R_5 : 3R_3), R_3)$	1	$(13, 7), (17, 3)$
20	$(R_{13} : (R_7 : R_3), R_3)$	1	$(13, 7), (17, 3)$
20	$(R_{13} : (R_5 : R_3), R_5)$	1	$(13, 7), (15, 5)$
20	$(R_{13} : (R_5 : 2R_3), 2R_3)$	1	$(13, 7), (14, 6)$
20	$(R_{13} : R_7, 2R_3)$	1	$(10, 10)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
20	$(R_{13} : 2R_5, R_3)$	1	$(10, 10)$
20	$(R_{13} : (R_5 : R_3), 3R_3)$	1	$(13, 7), (11, 9)$
20	$(R_{13} : R_5, 4R_3)$	1	$(12, 8)$
20	$(R_{13} : 7R_3)$	1	$(14, 6)$
20	$(R_{17} : R_5)$	1	$(16, 4)$
20	$(R_{17} : 3R_3)$	1	$(14, 6)$
20	$(R_{19} : R_3)$	1	$(18, 2)$
21	$(R_7 : 2(R_5 : 4R_3))$	1	$(14, 7), (21, 0)$
21	$(R_7 : (R_5 : 4R_3), (R_5 : 3R_3), R_3)$	1	$(18, 3), (11, 10), (14, 7)$
21	$(R_7 : (R_5 : 4R_3), (R_5 : R_3), R_5)$	1	$(16, 5), (12, 9), (14, 7)$
21	$(R_7 : (R_5 : 3R_3), (R_5 : 2R_3), R_5)$	1	$(14, 7), (11, 10), (12, 9), (13, 8)$
21	$(R_7 : (R_5 : 3R_3), 2(R_5 : R_3))$	1	$(11, 10), (16, 5), (18, 3), (14, 7),$ $(12, 9)$
21	$(R_7 : 2(R_5 : 2R_3), (R_5 : R_3))$	1	$(16, 5), (14, 7), (15, 6), (13, 8),$ $(12, 9)$
21	$(R_7 : (R_5 : 4R_3), (R_5 : 2R_3), 2R_3)$	1	$(15, 6), (14, 7), (13, 8)$
21	$(R_7 : 2(R_5 : 3R_3), 2R_3)$	1	$(11, 10), (14, 7), (15, 6)$
21	$(R_7 : (R_5 : 4R_3), 2R_5, R_3)$	1	$(11, 10), (17, 4)$
21	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), R_5, R_3)$	1	$(14, 7), (11, 10), (12, 9), (13, 8)$
21	$(R_7 : 2(R_5 : 2R_3), R_5, R_3)$	1	$(11, 10), (12, 9)$
21	$(R_7 : (R_5 : 2R_3), 2(R_5 : R_3), R_3)$	1	$(11, 10), (14, 7), (15, 6), (13, 8),$ $(12, 9)$
21	$(R_7 : (R_5 : 2R_3), 3R_5)$	1	$(15, 6), (14, 7)$
21	$(R_7 : 2(R_5 : R_3), 2R_5)$	1	$(14, 7), (11, 10), (12, 9)$
21	$(R_7 : (R_5 : 4R_3), (R_5 : R_3), 3R_3)$	1	$(16, 5), (14, 7), (12, 9)$
21	$(R_7 : (R_5 : 3R_3), (R_5 : 2R_3), 3R_3)$	1	$(11, 10), (14, 7), (12, 9), (13, 8)$
21	$(R_7 : (R_5 : 3R_3), 2R_5, 2R_3)$	1	$(13, 8), (17, 4)$
21	$(R_7 : (R_5 : 2R_3), (R_5 : R_3), R_5, 2R_3)$	1	$(14, 7), (11, 10), (12, 9), (13, 8)$
21	$(R_7 : 3(R_5 : R_3), 2R_3)$	1	$(14, 7), (11, 10), (12, 9), (13, 8)$
21	$(R_7 : (R_5 : R_3), 3R_5, R_3)$	1	$(15, 6), (17, 4)$
21	$(R_7 : (R_5 : 4R_3), R_5, 4R_3)$	1	$(19, 2), (12, 9)$
21	$(R_7 : (R_5 : 3R_3), (R_5 : R_3), 4R_3)$	1	$(16, 5), (11, 10), (14, 7), (12, 9)$
21	$(R_7 : 2(R_5 : 2R_3), 4R_3)$	1	$(14, 7), (12, 9), (13, 8)$
21	$(R_7 : (R_5 : 2R_3), 2R_5, 3R_3)$	1	$(16, 5), (17, 4)$
21	$(R_7 : 2(R_5 : R_3), R_5, 3R_3)$	1	$(16, 5), (14, 7), (12, 9)$
21	$(R_7 : 4R_5, 2R_3)$	1	$(20, 1)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15, 6), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15, 6), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_3^1 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_3^1 : ((R_5 : 2R_3) \oplus R_2), (R_5 : R_3))$	1	$(19, 2)$

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15, 6), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), (R_5 : R_3))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 4R_3), (R_5 : R_3))$	1	$(15, 6), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_{30}^{13} : ((R_5 : 2R_3) \oplus R_2), (R_5 : R_3))$	1, 2	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 4R_3), (R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{15}^2 : ((R_5 : R_3) \oplus R_2), (R_5 : 2R_3))$	1, 2	$(16, 5), (18, 3)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 \oplus R_3), (R_5 : 2R_3))$	2	$(13, 8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(19, 2), (15, 6), (16, 5), (20, 1)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^7 : ((R_5 : R_3) \oplus R_2), (R_5 : 2R_3))$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_3^1 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_3^1 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_3^1 : ((R_5 : R_3) \oplus R_2), (R_5 : 2R_3))$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(19, 2), (15, 6), (16, 5), (20, 1)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^{13} : ((R_5 : R_3) \oplus R_2), (R_5 : 2R_3))$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), (R_5 : 2R_3))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), (R_5 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{15}^7 : ((R_5 : R_3) \oplus R_2), (R_5 : 2R_3))$	1, 2	$(18, 3), (16, 5)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 \oplus R_3), (R_5 : 2R_3))$	2	$(13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (17, 4)$
21	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), 2R_5)$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (17, 4)$
21	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (17, 4)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), 2R_5)$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), (R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (17, 4)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 4R_3), 2(R_3 \oplus R_2))$	1	$(11, 10)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (13, 8), (12, 9), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_5 : R_3), R_5)$	1	$(16, 5), (12, 9)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(17, 4), (13, 8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_3), R_5)$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_{30}^7 : ((R_5 : R_3) \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(14, 7), (11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (13, 8), (12, 9), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_5 : R_3), R_5)$	1	$(16, 5), (12, 9)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(17, 4), (13, 8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_3), R_5)$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (13, 8), (12, 9), (15, 6)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10), (14, 7)$
21	$(R_7 : 1 + \nu_{30}^{13} : ((R_5 : R_3) \oplus R_2), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), (R_3 \oplus R_3), (R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^1 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6), (14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^2 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^1 : 2(R_5 : 2R_3), R_5)$	1	$(16, 5), (15, 6), (14, 7)$
21	$(R_7 : 1 + \nu_5^1 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15, 6), (16, 5), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^7 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6), (14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6), (14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_5 : 2R_3), R_5)$	1	$(14, 7), (15, 6), (16, 5)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15, 6), (16, 5), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(11, 10), (15, 6), (14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^7 : 2(R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 \oplus R_2), (R_5 : 2R_3), (R_3 \oplus R_2))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), 2(R_5 : R_3))$	1	$(11, 10), (15, 6), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14, 7), (12, 9)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(15, 6), (16, 5)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14, 7), (13, 8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11, 10), (12, 9)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,10),(12,9)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), 2(R_5 : R_3))$	1	$(11,10),(15,6),(13,8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14,7),(12,9)$
21	$(R_7 : 1 + \nu_3^1 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(19,2)$
21	$(R_7 : 1 + \nu_3^1 : (R_5 \oplus R_2), 2(R_5 : R_3))$	1	$(17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,10),(12,9)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), 2(R_5 : R_3))$	1	$(11,10),(15,6),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14,7),(12,9)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(15,6),(16,5)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14,7),(13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11,10),(12,9)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), 2(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(11,10),(12,9)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), 2(R_5 : R_3))$	1	$(11,10),(15,6),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_3 \oplus R_3), (R_5 : R_3))$	1	$(14,7),(12,9)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3), 2(R_3 \oplus R_3))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(14,7),(17,4),(13,8)$
21	$(R_7 : 1 + \nu_{15}^1 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), 3R_5)$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11,10),(14,7)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(15,6)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(13,8),(14,7),(17,4)$
21	$(R_7 : 1 + \nu_{15}^4 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(14,7),(17,4),(13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), 3R_5)$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11,10),(14,7)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(15,6)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10),(14,7),(17,4),(13,8)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 3R_3), 3(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10),(14,7),(15,6),(13,8),$ $(12,9)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10),(14,7),(13,8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 \oplus R_2), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10),(14,7),(15,6),(12,9),$ $(17,4)$
21	$(R_7 : 1 + \nu_{15}^2 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3), 2R_5)$	1	$(11,10),(12,9)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(14,7),(13,8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), 2R_5)$	1	$(11,10),(12,9)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (12,9)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (13,8),$ $(12,9)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (13,8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 \oplus R_2), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (12,9),$ $(17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (13,8),$ $(12,9)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 \oplus R_2), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (12,9),$ $(17,4)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_5 : R_3), 2R_5)$	1	$(11,10), (12,9)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_5 : R_3), (R_3 \oplus R_2), R_5)$	1	$(12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(14,7), (13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), 2R_5)$	1	$(11,10), (12,9)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), (R_3 \oplus R_2), R_5)$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (12,9)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (13,8),$ $(12,9)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 \oplus R_2), (R_5 : R_3), 2(R_3 \oplus R_2))$	1	$(11,10), (14,7), (15,6), (12,9),$ $(17,4)$
21	$(R_7 : 1 + \nu_{15}^7 : (R_5 : 2R_3), (R_3 \oplus R_3), 2(R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_{30}^1 : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (15,6), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(14,7), (11,10), (12,9)$
21	$(R_7 : 1 + \nu_{30}^1 : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (13,8)$
21	$(R_7 : 1 + \nu_5^1 : 3(R_5 : R_3), R_5)$	1	$(12,9)$
21	$(R_7 : 1 + \nu_5^1 : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 2(R_5 : R_3), R_5)$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_3), (R_5 : R_3), R_5)$	1	$(13,8)$
21	$(R_7 : 1 + \nu_5^1 : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12,9)$
21	$(R_7 : 1 + \nu_5^1 : 3(R_3 \oplus R_3), R_5)$	1	$(15,6)$
21	$(R_7 : 1 + \nu_{30}^7 : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (15,6), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(14,7), (11,10), (12,9)$
21	$(R_7 : 1 + \nu_{30}^7 : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (15,6), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (14,7), (12,9)$
21	$(R_7 : 1 + \nu_{30}^{11} : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (13,8)$
21	$(R_7 : 1 + \nu_5^2 : 3(R_5 : R_3), R_5)$	1	$(12,9)$
21	$(R_7 : 1 + \nu_5^2 : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_5 : R_3), R_5)$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), (R_5 : R_3), R_5)$	1	$(13,8)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(12,9)$
21	$(R_7 : 1 + \nu_5^2 : 3(R_3 \oplus R_3), R_5)$	1	$(15,6)$
21	$(R_7 : 1 + \nu_{30}^{13} : 3(R_5 : R_3), (R_3 \oplus R_2))$	1	$(11,10), (15,6), (12,9), (13,8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), 2(R_5 : R_3), (R_3 \oplus R_2))$	1	$(14, 7), (11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{30}^{13} : 2(R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2))$	1	$(11, 10), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{1} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (13, 8),$ $(17, 4)$
21	$(R_7 : 1 + \nu_{15}^{2} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 4R_5)$	1	$(14, 7), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), (R_3 \oplus R_2), 3R_5)$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 3(R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{30}^{7} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (13, 8),$ $(17, 4)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (13, 8),$ $(17, 4)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 4R_5)$	1	$(14, 7), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), (R_3 \oplus R_2), 3R_5)$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 2(R_3 \oplus R_2), 2R_5)$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 3(R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (12, 9)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (13, 8),$ $(17, 4)$
21	$(R_7 : 1 + \nu_{15}^{7} : (R_5 : 2R_3), 4(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{30}^{1} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (15, 6), (13, 8),$ $(12, 9), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{1} : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(14, 7), (11, 10), (16, 5), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), 3R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), 3R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{5}^{1} : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_3 \oplus R_3), 3R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_3 \oplus R_3), (R_3 \oplus R_2), 2R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_{5}^{1} : 2(R_3 \oplus R_3), 2(R_3 \oplus R_2), R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{30}^{7} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (15, 6), (13, 8),$ $(12, 9), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{7} : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(16, 5), (11, 10), (14, 7), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^{11} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (15, 6), (13, 8),$ $(12, 9), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(14, 7), (11, 10), (16, 5), (13, 8)$
21	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), 3R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{2} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), (R_5 : R_3), 3R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{5}^{2} : (R_3 \oplus R_3), (R_5 : R_3), (R_3 \oplus R_2), 2R_5)$	1	$(14, 7)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3), 2(R_3 \oplus R_2), R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), 3R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), (R_3 \oplus R_2), 2R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_5^2 : 2(R_3 \oplus R_3), 2(R_3 \oplus R_2), R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{30}^{13} : 2(R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(11, 10), (14, 7), (15, 6), (13, 8),$ $(12, 9), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_3 \oplus R_3), (R_5 : R_3), 3(R_3 \oplus R_2))$	1	$(14, 7), (11, 10), (16, 5), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^1 : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (19, 2),$ $(13, 8), (17, 4)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 5R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), (R_3 \oplus R_2), 4R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 2(R_3 \oplus R_2), 3R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 3(R_3 \oplus R_2), 2R_5)$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 4(R_3 \oplus R_2), R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^1 : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 5R_5)$	1	$(19, 2)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), (R_3 \oplus R_2), 4R_5)$	1	$(18, 3)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 2(R_3 \oplus R_2), 3R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 3(R_3 \oplus R_2), 2R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_5^1 : (R_3 \oplus R_3), 4(R_3 \oplus R_2), R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{30}^7 : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (19, 2),$ $(13, 8), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^{11} : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (19, 2),$ $(13, 8), (17, 4)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 5R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), (R_3 \oplus R_2), 4R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 2(R_3 \oplus R_2), 3R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 3(R_3 \oplus R_2), 2R_5)$	1	$(14, 7)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 4(R_3 \oplus R_2), R_5)$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_5^2 : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(12, 9)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 5R_5)$	1	$(19, 2)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), (R_3 \oplus R_2), 4R_5)$	1	$(18, 3)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 2(R_3 \oplus R_2), 3R_5)$	1	$(17, 4)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 3(R_3 \oplus R_2), 2R_5)$	1	$(16, 5)$
21	$(R_7 : 1 + \nu_5^2 : (R_3 \oplus R_3), 4(R_3 \oplus R_2), R_5)$	1	$(15, 6)$
21	$(R_7 : 1 + \nu_{30}^{13} : (R_5 : R_3), 5(R_3 \oplus R_2))$	1	$(11, 10), (16, 5), (14, 7), (19, 2),$ $(13, 8), (17, 4)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : 6(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : 5(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : 4(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : 3(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : 2(R_5 : R_3))$	1	$(12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_{30}^1 : (R_5 : R_3))$	1	$(13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : 6(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : 5(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : 4(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : 3(R_5 : R_3))$	1	$(11, 10), (12, 9), (13, 8)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : 2(R_5 : R_3))$	1	$(12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^7 + \nu_5^1 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : 6(R_5 : R_3))$	1	$(18,3), (16,5), (20,1), (15,6),$ $(19,2), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : 5(R_5 : R_3))$	1	$(18,3), (16,5), (15,6), (19,2),$ $(17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : 4(R_5 : R_3))$	1	$(18,3), (15,6), (16,5), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : 3(R_5 : R_3))$	1	$(15,6), (16,5), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : 2(R_5 : R_3))$	1	$(15,6), (16,5)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_{30}^1 : (R_5 : R_3))$	1	$(15,6)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : 6(R_5 : R_3))$	1	$(18,3), (16,5), (20,1), (15,6),$ $(19,2), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : 5(R_5 : R_3))$	1	$(18,3), (16,5), (15,6), (19,2),$ $(17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : 4(R_5 : R_3))$	1	$(18,3), (15,6), (16,5), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : 3(R_5 : R_3))$	1	$(15,6), (16,5), (17,4)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : 2(R_5 : R_3))$	1	$(15,6), (16,5)$
21	$(R_7 : 1 + \nu_{30}^{11} + \nu_3^1 : (R_5 : R_3))$	1	$(15,6)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : 6(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : 5(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : 4(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : 3(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : 2(R_5 : R_3))$	1	$(12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^1 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : 6(R_5 : R_3))$	1	$(16,5), (18,3), (20,1), (19,2),$ $(15,6), (17,4)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : 5(R_5 : R_3))$	1	$(16,5), (18,3), (20,1), (19,2),$ $(17,4)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : 4(R_5 : R_3))$	1	$(18,3), (19,2), (20,1), (17,4)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : 3(R_5 : R_3))$	1	$(18,3), (19,2), (20,1)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : 2(R_5 : R_3))$	1	$(19,2), (20,1)$
21	$(R_7 : 1 + \nu_5^2 + \nu_5^1 : (R_5 : R_3))$	1	$(20,1)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : 6(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : 5(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : 4(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : 3(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : 2(R_5 : R_3))$	1	$(12,9), (13,8)$
21	$(R_7 : 1 + \nu_5^2 + \nu_{30}^{11} : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : 6(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : 5(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : 4(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : 3(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : 2(R_5 : R_3))$	1	$(12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^1 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : 6(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : 5(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : 4(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : 3(R_5 : R_3))$	1	$(11,10), (12,9), (13,8)$

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : 2(R_5 : R_3))$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^1 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : 6(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : 5(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : 4(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : 3(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : 2(R_5 : R_3))$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_{30}^7 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : 6(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : 5(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : 4(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : 3(R_5 : R_3))$	1	$(11,10),(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : 2(R_5 : R_3))$	1	$(12,9),(13,8)$
21	$(R_7 : 1 + \nu_{30}^{13} + \nu_5^2 : (R_5 : R_3))$	1	$(13,8)$
21	$(R_{11} : (R_7 : 5R_3))$	1	$(12,9),(20,1)$
21	$(R_{11} : (R_7 : (R_5 : 2R_3)))$	1	$(19,2),(12,9),(13,8),(20,1)$
21	$(R_{11} : (R_7 : R_3,(R_5 : R_3)))$	1	$(19,2),(15,6),(17,4),(13,8)$
21	$(R_{11} : (R_7 : 2R_3,R_5))$	1	$(18,3),(14,7)$
21	$(R_{11} : (R_7 : 4R_3),R_3)$	1	$(12,9),(17,4)$
21	$(R_{11} : (R_7 : (R_5 : R_3)),R_3)$	1	$(19,2),(11,10),(12,9),(17,4)$
21	$(R_{11} : (R_7 : R_5,R_3),R_3)$	1	$(15,6),(14,7)$
21	$(R_{11} : (R_5 : 4R_3),R_5)$	1	$(11,10),(17,4)$
21	$(R_{11} : (R_7 : 2R_3),R_5)$	1	$(14,7),(13,8)$
21	$(R_{11} : (R_5 : 3R_3),(R_5 : R_3))$	1	$(19,2),(15,6),(17,4),(13,8)$
21	$(R_{11} : (R_7 : R_3),(R_5 : R_3))$	1	$(19,2),(15,6),(17,4),(13,8)$
21	$(R_{11} : 2R_7)$	1	$(12,9)$
21	$(R_{11} : 2(R_5 : 2R_3))$	1	$(15,6),(16,5),(17,4)$
21	$(R_{11} : R_7,(R_5 : 2R_3))$	1	$(12,9),(13,8)$
21	$(R_{11} : (R_7 : 3R_3),2R_3)$	1	$(14,7),(12,9)$
21	$(R_{11} : (R_7 : R_5),2R_3)$	1	$(14,7),(12,9)$
21	$(R_{11} : (R_5 : 3R_3),R_5,R_3)$	1	$(11,10),(14,7)$
21	$(R_{11} : (R_7 : R_3),R_5,R_3)$	1	$(11,10),(14,7)$
21	$(R_{11} : (R_5 : 2R_3),(R_5 : R_3),R_3)$	1	$(16,5),(15,6),(14,7),(13,8)$
21	$(R_{11} : R_7,(R_5 : R_3),R_3)$	1	$(11,10),(12,9)$
21	$(R_{11} : (R_5 : R_3),2R_5)$	1	$(11,10),(12,9)$
21	$(R_{11} : (R_5 : 4R_3),3R_3)$	1	$(15,6),(13,8)$
21	$(R_{11} : (R_7 : 2R_3),3R_3)$	1	$(11,10),(12,9)$
21	$(R_{11} : (R_5 : 2R_3),R_5,2R_3)$	1	$(11,10)$
21	$(R_{11} : R_7,R_5,2R_3)$	1	$(14,7)$
21	$(R_{11} : 2(R_5 : R_3),2R_3)$	1	$(11,10),(15,6),(13,8)$
21	$(R_{11} : 3R_5,R_3)$	1	$(14,7)$
21	$(R_{11} : (R_5 : 3R_3),4R_3)$	1	$(12,9),(13,8)$
21	$(R_{11} : (R_7 : R_3),4R_3)$	1	$(12,9),(13,8)$
21	$(R_{11} : (R_5 : R_3),R_5,3R_3)$	1	$(11,10),(13,8)$
21	$(R_{11} : (R_5 : 2R_3),5R_3)$	1	$(12,9),(13,8)$
21	$(R_{11} : R_7,5R_3)$	1	$(16,5)$
21	$(R_{11} : 2R_5,4R_3)$	1	$(16,5)$

Continued on next page

Table 2 – *Continued from previous page*

Weight	Type	Heights	Parities
21	$(R_{11} : (R_5 : R_3), 6R_3)$	1	$(15,6), (13,8)$
21	$(R_{11} : R_5, 7R_3)$	1	$(18,3)$
21	$(R_{11} : 10R_3)$	1	$(20,1)$
21	$(R_{13} : (R_7 : 3R_3))$	1	$(18,3), (16,5)$
21	$(R_{13} : (R_7 : R_5))$	1	$(18,3), (16,5)$
21	$(R_{13} : (R_5 : 4R_3), R_3)$	1	$(19,2), (12,9)$
21	$(R_{13} : (R_7 : 2R_3), R_3)$	1	$(15,6), (16,5)$
21	$(R_{13} : (R_5 : 2R_3), R_5)$	1	$(15,6), (14,7)$
21	$(R_{13} : R_7, R_5)$	1	$(11,10)$
21	$(R_{13} : 2(R_5 : R_3))$	1	$(19,2), (15,6), (17,4)$
21	$(R_{13} : (R_5 : 3R_3), 2R_3)$	1	$(16,5), (12,9)$
21	$(R_{13} : (R_7 : R_3), 2R_3)$	1	$(16,5), (12,9)$
21	$(R_{13} : (R_5 : R_3), R_5, R_3)$	1	$(14,7), (12,9)$
21	$(R_{13} : (R_5 : 2R_3), 3R_3)$	1	$(12,9), (13,8)$
21	$(R_{13} : R_7, 3R_3)$	1	$(12,9)$
21	$(R_{13} : 2R_5, 2R_3)$	1	$(12,9)$
21	$(R_{13} : (R_5 : R_3), 4R_3)$	1	$(11,10), (12,9)$
21	$(R_{13} : R_5, 5R_3)$	1	$(14,7)$
21	$(R_{13} : 8R_3)$	1	$(16,5)$
21	$(R_{17} : (R_5 : R_3))$	1	$(18,3), (20,1)$
21	$(R_{17} : R_5, R_3)$	1	$(15,6)$
21	$(R_{17} : 4R_3)$	1	$(13,8)$
21	$(R_{19} : 2R_3)$	1	$(17,4)$

Appendix B. Python code

This section gives the python code for the algorithms described in Section 4. The code contains the files

(a) `support.py` contains a number of useful functions for working with sorou and types;

(b) `ReferenceTypes.py` has the known types of weight ≤ 12 hardcoded and resets the type list `types.txt`;

(c) `TypeGen.py` loads the previous types list types.txt and adds the complete set of types with weight one greater than the previous types to `types.txt`;

(d) `types.txt` contains the current set of types;

(e) `sorouGenerator.py` contains code used in generating all sorou of a given weight as well as checking parities and height;

(f) `TypeMap.txt` contains a python dictionary with types as keys to a list of all sorou of the key's type;

(g) `WriteTypeList.py` reads the type list in `types.txt` and writes a csv file (`types.csv`) of the types formatted as in the Table 2 in Appendix A;

with each later file depending only on all of its precedents, except for `TypeGen.py` which also depends on `types.txt`.

B.1. support.py

```
""" Class Sorou:

    [ [ o_1, p_1 ], ..., [ o_j, p_j ] ]
    \nu_{o_1}^{p_1} + ... + \nu_{o_j}^{p_j}
"""


""" Class CommonRoot:

    [ o, p_1, ... p_j ]
    \nu_o^{p_1} + ... + \nu_o^{p_j}
"""


""" Class SubSorous:

    [ prime, [ p_1, f_1 ], ..., [ p_j, f_j ] ]
    \sum_{i = 1}^{j} \nu_prime^{p_i} \times f_i
"""


""" Class Type:

    [[ p, f_0, T_1, ... T_j ], ..., [ q, g_0, S_1, ... S_j ]]
    ( R_p : f_0 : T_1, ..., T_j ) (+) ( R_q : g_0 : S_1, ... S_j )
"""
```

```python
import math
import cmath
import fractions
import pickle
import pdb
import itertools
import time

from sympy import minimal_polynomial, sqrt, solve, QQ, I, simplify, expand
from sympy.functions import re, im
from sympy.abc import x, y

vanishingLimit = (0.1) ** (6)

q = { 1: [[1]] } # Used for partition generation

def decompose(n):
    try:
        return q[n]
    except:
        pass

    result = [[n]]

    for i in range(1, n):
        a = n-i
        R = decompose(i)
        for r in R:
            if r[0] <= a:
                result.append([a] + r)

    q[n] = result
    return result

def filterToLength (partitions, l):
    output = []
    for item in partitions:
        if len(item) == l:
            output += [item]
    return output

def filterToTwo (aPartition):
    output = []
    for item in aPartition:
        if 1 not in item and 2 in item:
            output += [item]
    return output
```

```python
def filterP (aPartition, l):
    output = filterToLength(aPartition, l)
    return output

def makepart (n, l):
    output = filterP(decompose(n),l)
    return output

def reverseList (aList):
    output = []
    for index in range(len(aList)):
        output.append(aList[-index - 1])
    return output

def isPrime(number):
    if number > 1:
        if number == 2:
            return True
        if number % 2 == 0:
            return False
        for current in range(3, int(math.sqrt(number) + 1), 2):
            if number % current == 0:
                return False
        return True
    return False

def getPrimes(number):
    while True:
        if isPrime(number):
            yield number
        number += 1

def findNthPrime(n):
    if n == 0:
        return 1

    primeGenerator = getPrimes(1)

    for index in range(n - 1):
        next(primeGenerator)

    return next(primeGenerator)

def findNtoMPrimes(n, m):

    output = []

    primeGenerator = getPrimes(1)
```

```python
    for index in range(n - 1):
        next(primeGenerator)

    for index in range(m - n):
        output.append(next(primeGenerator))

    return output

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def lcm(a, b):
    value = a * b // gcd(a, b)
    return value

def lcmm(args):
    tempLcm = 1
    index = 0
    while index < len(args):
        tempLcm = lcm(tempLcm, args[index])
        index += 1
    return tempLcm

def findMaxPrime (aWeightLimit):
    index = 0
    while findNthPrime(index + 1) < aWeightLimit:
        index += 1
    return findNthPrime(index)

def findMaxPrimeIndex (aWeightLimit):
    ps = findMaxPrime(aWeightLimit)
    index = 0
    while findNthPrime(index) != ps:
        index += 1
    if findNthPrime(index) == ps:
        return index
    else:
        return False

def pFactor (n):
    pFactors = []

    prime = 1
    primes = getPrimes(1)
```

```python
    while prime <= n:
        if n % prime == 0:
            pFactors.append(prime)
        prime = next(primes)

    return pFactors


def containsEquiPar (aParList):
    for par in aParList:
        if par[0] == par[1]:
            return True
    return False


def toRoot (z):
    if abs(z - 1) > vanishingLimit:
        print("This number is not a root of Unity")
        return False
    else:
        phi = cmath.phase(z) / (2 * cmath.pi)
        ratio = fractions.from_float(phi)
        return simplifyRoot(ratio.numerator, ratio.denominator)


def fromRoot (aRoot):
    z = cmath.exp(2*cmath.pi*aRoot[1]*1j / aRoot[0])
    return z


def simplifyRoot (aOmega, aPower):
    tempOmega = aOmega
    tempPower = aPower % aOmega

    newOmega = tempOmega // gcd(tempOmega, tempPower)
    newPower = tempPower // gcd(tempOmega, tempPower)

    return [newOmega, newPower]


def multiplyRoots (aOmega1, aPower1, aOmega2, aPower2):
    tempOmega = aOmega1 * aOmega2
    tempPower = aPower1 * aOmega2 + aOmega1 * aPower2
    return simplifyRoot(tempOmega, tempPower)


def findRootOrder (aRoot):
    flag = False
    index = 1
    while not flag:
        if index * aRoot[1] % aRoot[0] == 0:
            flag = True
            return index
        index += 1
```

```python
        if index > 1000000000:
            print("Order too high")
            return False
    print("Encountered some error")
    return False


def findOrder (aSorou):
    orders = []
    for item in aSorou:
        orders.append(findRootOrder(item))
    temp = set(orders)
    uniqueOrders = []
    for item in temp:
        uniqueOrders.append(item)
    return lcmm(uniqueOrders)


def findRelativeOrder (aSorou):
    rootRatios = []
    for item in aSorou:
        rootRatios.append(item)
        for other in aSorou:
            newRoot = [item[0] * other[0], item[0] * other[1] - item[1] * other[0]]
            rootRatios.append(newRoot)
    return findOrder(rootRatios)


def rotate (aSorou, aOmega, aPower):
    output = []
    for item in aSorou:
        output.append(multiplyRoots(item[0],item[1],aOmega,aPower))
    return output


def rotateFor1 (aSorou):
    # Rotates a sorou so that it includes a 1
    firstRoot = aSorou[0]
    return rotate(aSorou, firstRoot[0], (firstRoot[0] - firstRoot[1]) %
        firstRoot[0])


def orderRoots (root1, root2):
    if root1[0] > root2[0]:
        return 1
    elif root2[0] > root1[0]:
        return 2
    elif root1[1] > root2[1]:
        return 1
    elif root2[1] > root1[1]:
        return 2
    else:
        return 0
```

```python
def ssort (aSorou):
    tempSorou = []
    for root in aSorou:
        tempSorou.append(simplifyRoot(root[0],root[1]))
    return sorted(tempSorou, key=lambda tup: (tup[0],tup[1]) )

def isEqual (Sorou1, Sorou2):
    if len(Sorou1) != len(Sorou2):
        return False

    sSorou1 = ssort(Sorou1)
    sSorou2 = ssort(Sorou2)

    for i in range(len(sSorou1)):
        root1 = simplifyRoot(sSorou1[i][0], sSorou1[i][1])
        root2 = simplifyRoot(sSorou2[i][0], sSorou2[i][1])
        if root1[0] != root2[0]:
            return False
        if root1[1] != root2[1]:
            return False

    return True

def isEquiv (Sorou1, Sorou2):
    relOrder1 = findRelativeOrder(Sorou1)
    relOrder2 = findRelativeOrder(Sorou2)
    relOrder = lcm(relOrder1, relOrder2)

    tempSorou = Sorou2
    for i in range(relOrder):
        if isEqual(Sorou1, tempSorou):
            return True
        else:
            tempSorou = rotate(tempSorou, relOrder, 1)

    return False

def genRP (aPrime):
    RP = []
    for i in range(aPrime):
        RP.append([aPrime,i])
    return RP

def genAllSubSorous (aSorou):
    output = []
    for root in aSorou:
        newItems = []
```

```python
        for item in output:
            newItems.append(item + [root])
        output.append([root])
        if newItems: output += newItems
    return output


def findWeightofSorou (aSorou):
    weight = len(aSorou)
    return weight


def findWeightofType (aTypeListSum):
    weight = 0
    for typelist in aTypeListSum:
        weightList = [typelist[0], findWeightofSorou(typelist[1])]
        # print("weightList: ", weightList)
        for subType in typelist[2:]:
            if len(subType) == 1 & len(subType[0]) == 2:
                weightList.append(subType[0][0])
            else:
                weightList.append(findWeightofType(subType))
        # print("weightList with SubTypes: ", weightList)
        weightPartition = []
        for item in weightList[2:]:
            weightPartition.append(item - weightList[1])
        while len(weightPartition) < typelist[0]:
            weightPartition.append(weightList[1])
        # print("weightPartition: ", weightPartition)
        weight += sum(weightPartition)
    return weight


def findWeightPart (aMinVanType):
    '''
    This takes a minimal vanishing type: [[p, f0, T1, T2, ...]]
    and returns it's weight partition : [a,b,c,...]
    '''
    output = []
    wf0 = len(aMinVanType[0][1])
    p = aMinVanType[0][0]

    for item in aMinVanType[0][2:]:
        w = findWeightofType(item)
        output.append(w - wf0)

    for index in range(p - len(aMinVanType[0][2:])):
        output.append(wf0)

    return output
```

```python
def findHeightofSorou (aSorou):
    if len(aSorou) == 0:
        return 0

    working = ssort(aSorou)

    listsOfHeights = [1]

    for index in range(1,len(working)):
        if working[index] == working[index-1]:
            listsOfHeights[-1] += 1
        else:
            listsOfHeights.append(1)

    return max(listsOfHeights)

def toCommonRoot (aSorou):
    """
    Takes a sorou [[o,p],...,[o,p]] and converts it to sum of powers of a common
        root

    Stored as a Common Root list [relorder, l_1, l_2, ..., l_n] = sum_{i=1}^n
        \nu_{relorder}^{l_i}
    """
    relOrder = findRelativeOrder(aSorou)
    output = [relOrder]
    for item in aSorou:
        factor = relOrder // item[0]
        output.append(item[1]*factor)

    return output

def fromCommonRoot (aCommonRoot):
    """
    Takes a Common Root list [relorder, l_1, l_2, ..., l_n] and returns a Sorou
        List [[o,p],...,[o,p]]
    """
    sorou = []
    relOrder = aCommonRoot[0]
    powers = aCommonRoot[1:]
    for power in powers:
        sorou.append([relOrder,power])
    return ssort(sorou)

def splitRoot (aRoot, aTopPrime, aOtherPrimes):
    """
    This takes a root w = v_relorder^a: [relorder, a]
```

```python
    And returns two roots with product w, one a power of v_topPrime and the other
        a power of v_otherPrimes
    Returns FALSE if error
    """
    temp = simplifyRoot(aRoot[0], aRoot[1])

    if temp[0] == aTopPrime:
        return (temp, [1,0])

    root1 = [ aTopPrime, 0 ]
    root2 = [ aOtherPrimes, 0 ]

    for a in range(aTopPrime):
        for b in range(aOtherPrimes):
            test = multiplyRoots(root1[0],root1[1],root2[0],root2[1])
            if test[0] == temp[0] and test[1] == temp[1]:
                return
                    (simplifyRoot(root1[0],root1[1]),simplifyRoot(root2[0],root2[1]))
            else:
                root2[1] += 1
        root1[1] += 1
    print("ERROR: Cannot split root")
    return False

def formatted (aCommonRoot):
    """
    Helper function for toSubSorous
    Takes a list of common roots and returns it in subsorou format.
    Only use for prime order common roots (returns False otherwise)
    """
    output = []

    if aCommonRoot[0] not in primes:
        return False

    rootMap = {x : [] for x in range(aCommonRoot[0])}

    for power in aCommonRoot[1:]:
        rootMap[power].append([1,0])

    for item in rootMap.keys():
        output.append([item, rootMap[item]])

    return output

def rotateSSorousW (ssorou):
    """
    Rotates the subsorou form [p, [o,f], ..., [o,f]] so that
```

```python
        w(f_0) < w(f_j) for all j

    """
    output = [ssorou[0]]
    working = ssorou[1:]

    minIndex = 0
    minWeight = findWeightofSorou(working[0][1])
    for index in range(len(working)):
        newWeight = findWeightofSorou(working[index][1])
        if (newWeight < minWeight and newWeight > 0) or minWeight == 0:
            minIndex = index
            minWeight = newWeight

    rotateOrder = ssorou[0] - working[minIndex][0]

    for index in range(len(working)):
        originalIndex = (index + rotateOrder) % ssorou[0]
        newSubSorou = [index, working[(index + rotateOrder) % ssorou[0]][1]]
        output.append(newSubSorou)

    return output

def rotateSSorous1 (ssorou):
    """
    rotates the subsorou form [p, [o,f], ..., [o,f]] so that

        1 -< f_0

    """
    output = [ssorou[0]]
    working = ssorou[1:]

    firstRoot = ssorou[1][1][0]

    rotateOmega = firstRoot[0]
    rotatePower = (firstRoot[0] - firstRoot[1]) % rotateOmega

    for subSorou in working:
        o = subSorou[0]
        newSubSorou = rotate(subSorou[1], rotateOmega, rotatePower)
        output.append([o, newSubSorou])

    return output

def rotateSubSorous (ssorou):
    temp = rotateSSorousW(ssorou)
```

```python
    return rotateSSorous1(temp)

def toSubSorous (aSorou):
    """
    Takes a sorou [[o,p],...,[o,p]] and converts it to the sum of products of
        roots of order p_s a sub sorous

    Stored as [p, [o,f], ..., [o,f]]
    """
    output = []
    working = toCommonRoot(aSorou)

    relOrder = working[0]
    powers = working[1:]

    thesePrimes = pFactor(relOrder)

    if len(thesePrimes) == 1:
        output.append(thesePrimes[0])
        temp = formatted(working)
        if temp: output += temp
        return output

    topPrime = thesePrimes[-1]

    output.append(topPrime)

    rootMap = {x: [] for x in range(topPrime)}

    rotators = []
    splits = []
    SubSorous = []

    # Splits out top prime roots
    for power in powers:
        root = simplifyRoot(relOrder, power)
        first, second = splitRoot(root, topPrime, relOrder // topPrime)
        rotators.append(first)
        splits.append(second)
    for index in range(len(rotators)):
        root = rotators[index]
        rootMap[root[1]] += [splits[index]]
    for index in range(topPrime):
        output.append([index, rootMap[index]])

    temp = rotateSubSorous(output)

    output2 = [output[0]]
```

```python
    for item in temp[1:]:
        if item[1]: output2.append(item)

    return output2

def printSubSorous (aSubSorou):
    print(aSubSorou[0])
    for item in aSubSorou[1:]:
        print(item)

def fromSubSorous (ssorou):
    topPrime = ssorou[0]
    output = []

    for item in ssorou[1:]:
        for item2 in item[1]:
            newroot = multiplyRoots( topPrime, item[0], item2[0], item2[1] )
            output.append(newroot)

    return output

def evalSorouN (aSorou):
    """
    Takes a Sorou list [[o,p],...,[o,p]] and returns the value as
    a complex number z
    """
    nums = []
    for root in aSorou:
        nums.append(cmath.exp(2*cmath.pi*root[1]*1j / root[0]))
    return sum(nums)

def isVanishingN (Sorou):
    value = evalSorouN(Sorou)
    if abs(value) < vanishingLimit:
        return True
    else:
        return False

def findPrimitiveRoot (n):
    if (n == 1):
        return solve(x - 1)[0]
    else:
        unorderedRoots = solve(x**n - 1)
        if unorderedRoots[0] != 1:
            currentMax = unorderedRoots[0]
        else:
            currentMax = unorderedRoots[1]
        for item in unorderedRoots:
```

```python
        if re(item) >= re(currentMax) and im(item) > 0:
            currentMax = item
    return currentMax

def raiseAlgebraicToPower(aPrimitiveRoot, aPower):
    output = 1

    for index in range(aPower):
        output = simplify(expand(output * aPrimitiveRoot))

    return output

def isVanishing (aSorou):
    algebraicRoots = []

    for root in aSorou:
        primitiveRoot = findPrimitiveRoot(root[0])
        algebraicRoots.append(simplify(expand(primitiveRoot ** root[1])))

    return sum(algebraicRoots) == 0

def isMinVan (aSorou):
    """
    Based on Prop 2.3, check if the sorou is minimal vanishing.
    """
    if not aSorou:
        return False

    working = toSubSorous(aSorou)
    SubSorous = working[1:]

    if not isVanishingN(aSorou):
        return False

    # First criteria
    if isVanishingN(working[1][1]):
        return False

    # Second & Third Criteria
    values = []
    for item in SubSorous:
        valsToAdd = set()
        allSubSorous = genAllSubSorous(item[1])[:-1]
        for subitem in allSubSorous:
            valsToAdd.add(evalSorouN(subitem))
            if isVanishingN(subitem):
                if len(subitem) != len(item[1]):
                    return False
```

```python
        values.append(valsToAdd)

    # Third Criteria
    if len(values[0]) == 0:
        return True

    intersection = values[0]
    for vals in values[1:]:
        intersection = intersection & vals

    if len(intersection) == 0:
        return True

    return False

def orderTypes (typeSum1, typeSum2):
    """
    Compares the types type1 and type2 and returns either 1 or 2 depending on
        lower order type.
    If the types are the same returns 0

        typeSum1 = [[[p, f0] + [t_1, ..., t_j]], ..., [...]]
        typeSum2 = [[[q, g0] + [s_1, ..., s_i]], ..., [...]]

        (with t_a and s_a listed in ascending order)

    Order is defined by type1 > type2 if and only if (in order or presidence):

        I.  Weight Type1 > Weight Type2
        I.  Number of minimal vanishing sorous of type1 > type2
        II. For minvan types listed in decending order:
            1.  p >= q
            2.  w(f0) > w(g0)
            3.  For roots e^{\phi_a i} in f0, e^{\theta_a i} in g0:
                A. \phi_a > \theta_a
            4.  j > i
            5.  For subTypes t_a, s_a:
                A.  t_a > s_a

    """
    if findWeightofType(typeSum1) > findWeightofType(typeSum2):
        return 1
    elif findWeightofType(typeSum2) > findWeightofType(typeSum1):
        return 2

    if len(typeSum1) > len(typeSum2):
        return 1
    elif len(typeSum2) > len(typeSum1):
```

```python
        return 2

    for index in range(len(typeSum1)):
        type1 = typeSum1[index]
        type2 = typeSum2[index]

        if type1 == type2:
            continue

        if type1[0] > type2[0]:
            return 1
        elif type2[0] > type1[0]:
            return 2

        f0 = type1[1]
        g0 = type2[1]
        wf0 = findWeightofSorou(f0)
        wg0 = findWeightofSorou(g0)

        if wf0 > wg0:
            return 1
        elif wg0 > wf0:
            return 2

        for sorouIndex in range(wf0):
            phi1 = f0[sorouIndex][1] / f0[sorouIndex][0]
            phi2 = g0[sorouIndex][1] / g0[sorouIndex][0]
            if phi1 > phi2:
                return 1
            elif phi2 > phi1:
                return 2

        if len(type1) == 2 and len(type2) == 2:
            continue
        elif len(type1) > len(type2):
            return 1
        elif len(type2) > len(type1):
            return 2

        for index in range(2,len(type1)):
            subTypeOrder = orderTypes(type1[index], type2[index])
            if subTypeOrder == 1:
                return 1
            elif subTypeOrder == 2:
                return 2

    return 0
```

```python
def subtractSorous (Sorou1, Sorou2):
    '''
    returns sorou1 - sorou2
    '''
    output = []
    sumand = ssort(Sorou1)
    negand = ssort(Sorou2)

    index1 = 0
    index2 = 0

    tempSumand = []
    tempNegand = []

    while index1 < len(sumand) and index2 < len(negand):
        root1 = sumand[index1]
        root2 = negand[index2]

        tester = orderRoots(root1, root2)
        if tester == 0:
            index1 += 1
            index2 += 1
        elif tester == 1:
            tempNegand.append(root2)
            index2 += 1
        else:
            tempSumand.append(root1)
            index1 += 1

    if index1 < len(sumand):
        tempSumand += sumand[index1:]
    if index2 < len(negand):
        tempNegand += negand[index2:]

    output += tempSumand
    output += rotate(tempNegand, 2, 1)

    return output

def subtractF0 (f0, aSubSorou):
    working = aSubSorou.copy()

    for index in range(len(aSubSorou)):
        tester = subtractSorous(f0, working)
        if len(tester) == len(aSubSorou) - len(f0):
            return tester

        temp = working[1:] + [working[0]]
```

```python
        rotateOmega = temp[0][0]
        rotatePower = (rotateOmega - temp[0][1]) % rotateOmega
        working = rotate(temp, rotateOmega, rotatePower)

    return False


def isSubSorou (aSorou1, aSorou2):
    # Checks if aSorou2 is contained within aSorou1
    working = subtractSorous(aSorou1, aSorou2)
    if len(working) == len(aSorou1) - len(aSorou2):
        return True
    else:
        return False


def makeWeightMap (aTypeList):
    theTypes = {}
    for T in aTypeList:
        w = findWeightofType(T)
        if w in theTypes.keys():
            theTypes[w] += [T]
        else:
            theTypes[w] = [T]
    maxWeight = sorted(theTypes.keys())[-1]
    for w in range(maxWeight):
        if w not in theTypes.keys():
            theTypes[w] = []
    return theTypes


def makeFullWeightMap (aWeightMap):
    weightLimit = sorted(aWeightMap.keys())[-1]

    output = {0:[], 1:[]}

    for index in range(2,weightLimit):
        newTypes = []
        # indexPartitions = filterToTwo(decompose(index))
        indexPartitions = decompose(index)
        for partition in indexPartitions:
            weightList = [aWeightMap[innerIndex] for innerIndex in partition]
            combos = [p for p in itertools.product(*weightList)]
            for item in combos:
                newType = []
                for aType in item:
                    newType += aType
                if checkTypesIsOrdered(item) and newType not in newTypes:
                    newTypes.append(newType)
        output[index] = newTypes
```

```python
        return output


def orderByWeight (aTypeList):
    output = []
    theTypes = makeWeightMap(aTypeList)
    for index in sorted(theTypes.keys()):
        output += theTypes[index]
    return output


def checkTypesIsOrdered (aTypeList):
    for index in range(1,len(aTypeList)):
        if orderTypes(aTypeList[index-1], aTypeList[index]) == 2:
            return False
    return True


def genFromTypeForF0 (SorouTypeSum, f0):
    """
    Generates a sorou for each minimial vanishing type and rotates them so that
        it contains f_0
    """
    sorouList = []
    for innerType in SorouTypeSum:
        minVanType = [innerType]
        temp = genFromType(minVanType)
        sorouList.append(rotateFor1(temp))

    if len(sorouList) < len(f0):
        return False

    output = []

    for index in range(len(f0)):
        rotateOmega = f0[index][0]
        rotatePower = f0[index][1]
        output += rotate(sorouList[index],rotateOmega, rotatePower)

    for subSorou in sorouList[len(f0):]:
        output += subSorou

    return output


def genFromType (SorouTypeSum):
    """
    Takes a type [[ p, f_0, T_1, T_2, .., T_j ], ... ]
    and returns a representative sorou with this type
    """
    sorou = []
```

```python
    for SorouType in SorouTypeSum:
        if len(SorouType) == 2:
            sorou += genRP(SorouType[0])
        else:
            subTypes = SorouType[2:]
            if len(subTypes) >= SorouType[0]: return False
            subSorous = [SorouType[1]]
            differences = []
            for subType in subTypes:
                if len(subType) == 1:
                    temp = genFromType(subType)
                else:
                    temp = genFromTypeForF0(subType, SorouType[1])
                if not temp: return False
                differences.append(ssort(temp))

            for item in differences:
                subSorou = subtractF0(SorouType[1], item)
                if not subSorou: return False
                subSorous.append(subSorou)

            tempLen = SorouType[0] - len(subTypes) - 1
            if tempLen:
                for index in range(tempLen):
                    subSorous.append(SorouType[1])

            sSorou = [SorouType[0]]
            for index in range(SorouType[0]):
                sSorou.append([index, subSorous[index]])

            sorou += fromSubSorous(sSorou)

    return ssort(sorou)

def findParity (aSorou):
    """
    Note that a root (-1)^n \nu_o^p in minimal terms can be rewritten
        +\nu_{o'}^{p}
    With o' % 2 == 0 if and only if n % 2 = 1
    """
    odds = 0
    evens = 0
    for root in aSorou:
        newRoot = simplifyRoot(root[0], root[1])
        if newRoot[0] % 2 == 0:
            evens += 1
        else:
            odds += 1
```

```python
        return (max(odds, evens), min(odds,evens))

def findTopPrimeIndex (aWeightLimit):
    index = 0
    while findNthPrime(index) < aWeightLimit:
        index += 1
    return index

def genP1MinVanTypeList (aWeightLimit):
    """
    generates the first types based on [primes] defined above
    All the ones based on R_p and then all the ones based on R_{p+1}
    returns a list of the types
    """

    n = findTopPrimeIndex(aWeightLimit)

    primeTypes = [ [[p, [[1,0]]]] for p in findNtoMPrimes(2,n) ]
    allTypes = [primeTypes[0]]

    for item in primeTypes[1:]:
        topPrime = item[0][0]
        toAdd = [[item]]

        for index in range(topPrime-1):
            innerToAdd = []
            previous = toAdd[index]

            for prevType in previous:
                for extraType in allTypes:
                    if len(prevType[0]) == 2:
                        newType = [prevType[0] + [extraType]]
                        w = findWeightofType(newType)
                        if w < aWeightLimit:
                            innerToAdd.append(newType)
                    elif orderTypes(extraType, prevType[0][-1]) in [0,2]:
                        newType = [prevType[0] + [extraType]]
                        w = findWeightofType(newType)
                        if w < aWeightLimit:
                            innerToAdd.append(newType)
            toAdd.append(innerToAdd)

        for innerList in toAdd:
            for subInnerList in innerList:
                allTypes.append(subInnerList)

    return allTypes
```

```python
def genVanTypes (aWeightLimit):
    output = []
    minVanTypes = genP1MinVanTypeList(aWeightLimit)

    vanillaTypeList = reverseList([[[2,[[1,0]]]]] + minVanTypes)

    for i in range(len(vanillaTypeList)):
        firstType = vanillaTypeList[i]
        toAdd = [[firstType]]
        for index in range(aWeightLimit - findWeightofType(firstType)):
            innerToAdd = []
            try:
                previous = toAdd[index]
            except:
                continue

            for prevType in previous:
                for extraType in vanillaTypeList[i:]:
                    if findWeightofType(prevType + extraType) < aWeightLimit:
                        innerToAdd.append(prevType + extraType)
            if len(innerToAdd) > 0: toAdd.append(innerToAdd)
        for inner in toAdd:
            output += inner

    return output


def checkForAMinVanType (aTypeList, minVanTypes):
    for aType in aTypeList:
        if aType in minVanTypes:
            return True
    return False


def reduceByPs (someTypes, aTopPrime):
    output = []
    for item in someTypes:
        if item[0][0] < aTopPrime:
            output.append(item)
    return output


def genPTypeList (aPartition, aTypeMap, aTopPrime, aWF0):
    output = []
    for index in aPartition:
        someTypes = aTypeMap[index+aWF0]
        filtered = reduceByPs(someTypes, aTopPrime)
        output.append(filtered)
    return output


def genSubTypeCombos (aPTypeList, minVanTypes):
```

```python
    output = []

    endIndex = len(aPTypeList)        # Note that this is the top prime
    listLens = []
    for item in aPTypeList:
        listLens.append(len(item))

    indexLists = []

    for i in range(endIndex):
        newIndexLists = []
        if i == 0:
            for index in range(listLens[i]):
                newIndexLists.append([index])
        else:
            for item in indexLists:
                for index in range(listLens[i]):
                    newIndexLists.append( item + [index] )

        indexLists = newIndexLists

    for l in indexLists:
        typeCombo = []
        for i in range(endIndex):
            typeCombo.append(aPTypeList[i][l[i]])
        if checkForAMinVanType(typeCombo, minVanTypes) and
            checkTypesIsOrdered(typeCombo):
            output.append(typeCombo)

    return list(output)

def isR2Sum (aType):
    if aType == [[2,[[1,0]]]]:
        return True
    if aType == [[2,[[1,0]]],[2,[[1,0]]]]:
        return True
    if aType == [[2,[[1,0]]],[2,[[1,0]]],[2,[[1,0]]]]:
        return True
    return False

def removeR2 (aSubTypeCombo):
    output = []
    for item in aSubTypeCombo:
        if not isR2Sum(item):
            output.append(item)
    return output

def prodPrevPrimes (aPrime):
```

```python
    index = 0
    output = 1
    while findNthPrime(index) < aPrime:
        output *= findNthPrime(index)
        index += 1
    return output


def genf0s (aWeight, aTopPrime):
    layerOutput = [[[1,0]]]
    relOrder = prodPrevPrimes(aTopPrime)
    for index in range(aWeight - 1):
        nextLayerOutput = []
        for item in layerOutput:
            if index == 0:
                for i in range(1,relOrder // 2):
                    nextLayerOutput.append(item + [[relOrder, i]])
            else:
                for i in range(1, item[-1][-1]):
                    nextLayerOutput.append(item + [[relOrder, i]])
        layerOutput = nextLayerOutput
    return layerOutput


def findPrimesToCheck(aWeight):
    # Note that we only consider primes stricly greater than 5
    # Because the maximum weight type with top prime 5 is
    # (R_5 : 4R_3) which is already hard coded into the reference types
    output = []
    primeGenerator = getPrimes(6)

    p = next(primeGenerator)
    while p <= aWeight:
        output.append(p)
        p = next(primeGenerator)

    return output


######## Functions for formatting output:


def printNestedList (aList, counter = 0):
    for item in aList:
        if type(item) != list:
            print(' ' * counter, item)
        elif type(item[0]) != list:
            print(' ' * counter, item)
        elif type(item[0][0]) != list:
            print(' ' * counter, item)
        else:
            print(' ' * counter, '[')
```

```python
        printNestedList(item, counter + 1)
        print(' ' * counter, ']')


def rootToLatexString (root):
    """
    Takes a root [a,b] and returns a string "\nu_{a}^{b}"
    """
    temp = simplifyRoot(root[0],root[1])
    if temp == [1,0]:
        return str(1)
    else:
        return "\\nu_{" + str(temp[0]) + "}^{" + str(temp[1]) + "}"


def sorouToLatexString (Sorou):
    """
    Takes a sorou:
    and returns a latex formatted string "\nu_o^p + ..."
    """
    output = ""
    for index in range(len(Sorou)):
        output += rootToLatexString(Sorou[index])
        if index != len(Sorou) - 1:
            output += "+"
    return output


def sorouToTikzString (aSorou):
    print('\\begin\{tikzpicture\}')
    print('\\draw[<->, gray] (-4,0) -- (4,0);')
    print('\\draw[<->, gray] (0,-4) -- (0,4);')
    print('\\draw[dashed, thin, gray] (0,0) circle (3);')

    for aRoot in aSorou:
        phi = (360 / aRoot[0]) * aRoot[1]
        psi = round(phi)
        colour = 'black'
        print('\\draw[fill = {}] (0,0) -- ({}:3) circle (0.7mm);'.format(colour,
            psi))

    print('\\end\{tikzpicture\}')


def npMinVanTypeToLatexString (typelist):
    """
    Takes a type list [ p, f_0 ] + [T_1, T_2, .., T_p ]
    and creates a string "(R_p : f_0 : T_1, ... T_j)"
    """
    output = "R_"

    output += '{' + str(typelist[0]) + '}'
```

```python
    f0str = sorouToLatexString(typelist[1])
    if f0str != "1": output += " : " + f0str

    if len(typelist) == 2:
        return output
    else:
        output += " : "

    index = 2
    counter = 1
    while index < len(typelist):
        curType = typelist[index]
        nextType = typelist[index + 1] if index + 1 < len(typelist) else 0

        if curType == nextType:
            counter += 1
            index += 1
        elif curType[0] != [2,[[1,0]]]:
            if counter > 1:
                output += str(counter) + typeToLatexString(curType)
            else:
                output += typeToLatexString(curType)

            if index != len(typelist) - 1:
                output += "; " # comma for latex, semi-colon for csvs
            index += 1
            counter = 1
        else:
            if index == len(typelist) - 1:
                output = output[:-2]
            index += 1
            counter = 1

    return output

def minVanTypeToLatexString (typelist):
    # Add brackets to npTypeString
    if len(typelist) == 2:
        return npMinVanTypeToLatexString(typelist)
    else:
        return "( " + npMinVanTypeToLatexString(typelist) + " )"

def typeToLatexString (typeListSum):
    output = ""
    if len(typeListSum) > 1:
        output = "("

    for innerType in typeListSum[:-1]:
```

```
            output += minVanTypeToLatexString(innerType) + ' \oplus '
        output += minVanTypeToLatexString(typeListSum[-1])

        if len(typeListSum) > 1:
            output += ")"

        return output

def tPrint (anArgs):

        output = time.asctime() + "; "

        for item in anArgs:
            output = output + str(item)

        print(output)

######## Build a full type list:

def main():

        genf0s(3,7)

        return 0

if __name__ == '__main__':
        main()
```

B.2. ReferenceTypes.py

```
import support
import pickle

#### Prime Types

R2 = [[2,[[1,0]]]]
R3 = [[3,[[1,0]]]]
R5 = [[5,[[1,0]]]]
R7 = [[7,[[1,0]]]]
R11 = [[11,[[1,0]]]]
R13 = [[13,[[1,0]]]]

#### Type Weight Dictionary
##   Key = Weight
##   Entry = List of minimal vanishing types of that weight

refTypesMap = {
```

```
0: [],

1: [],

2: [ R2 ],

3: [ R3 ],

4: [],

5: [ R5 ],

6: [
      [[5,[[1,0]], R3 ]]
   ],

7: [
      R7,
      [[5,[[1,0]], R3, R3 ]]
   ],

8: [
      [[7,[[1,0]], R3 ]],
      [[5,[[1,0]], R3, R3, R3 ]]
   ],

9: [
      [[7,[[1,0]], R3, R3 ]],
      [[5,[[1,0]], R3, R3, R3, R3 ]]
   ],

10: [
      [[7,[[1,0]], R5 ]],
      [[7,[[1,0]], R3, R3, R3 ]]
    ],

11: [
      R11,
      [[7,[[1,0]], [[5,[[1,0]], R3 ]] ]],
      [[7,[[1,0]], R5, R3 ]],
      [[7,[[1,0]], R3, R3, R3, R3 ]]
    ],

12: [
      [[11,[[1,0]], R3 ]],
      [[7,[[1,0]], [[5,[[1,0]], R3, R3 ]] ]],
      [[7,[[1,0]], R3, [[5,[[1,0]], R3 ]] ]],
      [[7,[[1,0]], R3, R3, R5 ]],
```

```
                  [[7,[[1,0]], R3, R3, R3, R3, R3]]
               ]
         }

def main ():

    print("Rewriting to reference type list")

    refTypeList = [ refTypesMap[i] for i in range(13) ]

    typeList = []

    for item in refTypeList:
        typeList += item


    with open('types.txt', 'wb') as rt:
        #### Reset the reference type list
        pickle.dump(typeList, rt)

    print("Finished rewrite")
    print("Written Types until weight:
        {}".format(support.findWeightofType(typeList[-1])))
    return 0

if __name__ == '__main__':
    main()
```

B.3. TypeGen.py

```
import support
import pickle
import pdb


def genNextTypes (aPrevTypes):
    """
    Returns a list of all minimal vanishing types with weight one more than the
        highest
    weight in aPrevTypes
    """

    output = []

    prevWeight = support.findWeightofType(aPrevTypes[-1])
    currentWeight = prevWeight + 1
```

```python
    baseTypeWeights = support.makeWeightMap(aPrevTypes)
    allTypeWeights = support.makeFullWeightMap(baseTypeWeights)

    primesToCheck = support.findPrimesToCheck(currentWeight)

    for prime in primesToCheck:
        partitions = support.makepart(currentWeight, prime)

        for partition in partitions:

            f0Possibilities = support.genf0s(partition[-1], prime)

            for f0 in f0Possibilities:

                pTypeList = support.genPTypeList(partition, allTypeWeights, prime,
                    len(f0))
                subTypeListCombinations = support.genSubTypeCombos(pTypeList,
                    aPrevTypes)

                for aSubTypeCombo in subTypeListCombinations:
                    trueSubTypeCombo = support.removeR2(aSubTypeCombo)
                    aNewType = [[prime, f0] + trueSubTypeCombo]
                    testSorou = support.genFromType(aNewType)

                    # if len(f0) > 1: pdb.set_trace()

                    if support.isMinVan(testSorou) and
                        support.findWeightofType(aNewType) == currentWeight:
                        output.append(aNewType)

    return output

def main():

    with open("types.txt", "rb") as tls:
        previousTypeList = pickle.load(tls)

    print('Starting Now')

    newTypes = genNextTypes(previousTypeList)

    allTypes = previousTypeList + newTypes

    with open("types.txt", "wb") as tls:
        pickle.dump(allTypes, tls)

    prevWeight = support.findWeightofType(previousTypeList[-1])
    currentWeight = prevWeight + 1
```

```python
    print('{0} Types Generated of weight {1}'.format(len(newTypes),
        currentWeight))

    print('--------------')
    for item in newTypes:
        print(support.findWeightofType(item) ,support.typeToLatexString(item))
    print('--------------')

    return 0

if __name__ == '__main__':
    main()
```

B.4. **sorouGenerator.py**

```python
import support
import pickle
import itertools
import ReferenceTypes as rt
import TypeGen
import pdb
import time
from multiprocessing import Pool as ThreadPool

def genBaseSorou (aMinVanType):
    working = aMinVanType[0]
    baseSorou = []
    for index in range(working[0]):
        baseSorou.append([support.rotate(working[1], working[0], index)])
    return baseSorou

#### https://stackoverflow.com/questions/6284396/permutations-with-unique-values
class unique_element:
    def __init__(self,value,occurrences):
        self.value = value
        self.occurrences = occurrences

#### https://stackoverflow.com/questions/6284396/permutations-with-unique-values
def perm_unique(elements):
    eset=set(elements)
    listunique = [unique_element(i,elements.count(i)) for i in eset]
    u=len(elements)
    return perm_unique_helper(listunique,[0]*u,u-1)

#### https://stackoverflow.com/questions/6284396/permutations-with-unique-values
def perm_unique_helper(listunique,result_list,d):
```

```python
    if d < 0:
        yield tuple(result_list)
    else:
        for i in listunique:
            if i.occurrences > 0:
                result_list[d]=i.value
                i.occurrences-=1
                for g in perm_unique_helper(listunique,result_list,d-1):
                    yield g
                i.occurrences+=1

def findSubTypePermutations (n, p, myPrevPerms):
    if n > p: return False

    if (n,p) in myPrevPerms.keys():
        return myPrevPerms[(n,p)]

    working = [0] * p

    if n == 0:
        mPerm = tuple(working)
        return set([mPerm])

    for index in range(n):
        working[index] = index + 1

    unfiltered = list(perm_unique(working[1:]))
    output = set()
    for item in unfiltered:
        output.add(tuple([1]+ list(item)))

    myPrevPerms[(n,p)] = output

    return output

def makeBitMap (p):
    output = []

    for index in range(2 ** p):
        bitsString = bin(index)[2:]
        bits = [0] * (p - len(bitsString))
        for char in bitsString:
            bits.append(int(char))
        output.append(bits)

    return output

def selectSubSorous (aBits, aNewSorou):
```

```python
    output = []

    for index in range(len(aBits)):
        output += aNewSorou[index][aBits[index]]

    return output

def shouldAdd (aSorou, aSorouSet):
    for item in aSorouSet:
        if support.isEquiv(aSorou, item):
            return False

    return True

def toStr(n, base):
    # http://interactivepython.org/courselib/static/pythonds/
    #       Recursion/pythondsConvertinganIntegertoaStringinAnyBase.html
    convertString = "0123456789ABCDEFGHIJK"
    if n < base:
        return convertString[n]
    else:
        return toStr(n // base, base) + convertString[n % base]

def toBaseN (number, base, length):
    inBase = toStr(number, base)

    toAdd = length - len(inBase)

    strToAdd = '0' * toAdd

    return strToAdd + inBase

def makeMap (m, n):
    if m > 19:
        print('Error, f_0 too long')

    output = []

    temp = [0] * m

    for index in range(n ** m):
        indexBaseM = list(toBaseN(index, n, m))
        indexArray = [int(item) for item in indexBaseM]
        output.append(indexArray)

    return output

def partitionF0 (aF0, n):
```

```python
    #### Partitions aF0 into n sub sorous
    #### Returns a list of all such paritions

    output = []

    mappings = makeMap(len(aF0), n)

    for mapping in mappings:
        temp = [[]] * n
        for index in range(len(aF0)):
            temp[mapping[index]] = temp[mapping[index]] + [aF0[index]]
        shouldAdd = True
        for item in temp:
            if item == []:
                shouldAdd = False
        if shouldAdd:
            output.append(temp)

    return output

def canBeSubSorou (aSorou, anotherSorou):
    """
    Returns true if anotherSorou can be rotated so that it is a subSorou of
        aSorou, and false otherwise
    """

    output = False

    aSorouRelOrder = support.findRelativeOrder(aSorou)
    anotherSorouRelORder = support.findRelativeOrder(anotherSorou)
    crossRelativeOrder = support.lcm(aSorouRelOrder, anotherSorouRelORder)

    for index in range(crossRelativeOrder):
        rotatedSorou = support.rotate(anotherSorou, crossRelativeOrder, index)
        if support.isSubSorou(aSorou, rotatedSorou):
            output = True
            break

    return output

def matchTypeToSorous (aF0Parition, aSorouListList):
    #### Output is a three dimensional array
    ##      Index 1 - The sorou in aF0Parition
    ##      Index 2 - The subtype index
    ##      Index 3 - The specific sorou of subtypes[I2]

    output = []
    flagList = []
```

```python
    for sorou in aF0Parition:
        t2 = []
        for aSorouList in aSorouListList:
            t1 = []
            for aSorou in aSorouList:
                if canBeSubSorou(aSorou, sorou):
                    t1.append(True)
                else:
                    t1.append(False)
            t2.append(t1)
        output.append(t2)

    return output

def simplifyArray (a3DBoolArray):
    rowN = len(a3DBoolArray)
    colN = len(a3DBoolArray[0])

    output = []

    for rowIndex in range(rowN):
        output.append([])
        for colIndex in range(colN):
            if (True in a3DBoolArray[rowIndex][colIndex]):
                output[rowIndex].append(True)
            else:
                output[rowIndex].append(False)

    return output

def findRotationIndicies (aSubSorou, aF0):
    output = set()

    for root in aSubSorou:
        a = root[0] * aF0[0][0]
        b = (root[0] * aF0[0][1] - aF0[0][0] * root[1]) % a
        output.add((a, b))

    return output

def checkF0OddParity (aF0):
    if len(aF0) == 1:
        return True
    elif aF0[0][1] % 2 == 0:
        return False
    else:
        return True
```

```python
def selectFromNewSorou (aNewSorou):
    output = [[]]
    for item in aNewSorou:
        temp = []
        for subItem in item:
            for i in output:
                temp.append(i+subItem)
        output = temp

    return output

def genAllSorousOfMinVanType (aMinVanType, aTypeSorouMap, aPermutationsMap):
    aTypeStr = support.typeToLatexString(aMinVanType)

    if aTypeStr in aTypeSorouMap.keys() and aTypeSorouMap[aTypeStr] != []:
        return aTypeSorouMap[aTypeStr]

    outputKeys = set()
    output = []

    subTypes = aMinVanType[0][2:]

    subSorous = []
    for subType in subTypes:
        subSorou = genAllSorousOfType(subType, aMinVanType[0][1], aTypeSorouMap,
            aPermutationsMap)
        subSorous.append(subSorou)

    support.tPrint("Generating SubType Permutations")
    subTypePermunations = findSubTypePermutations(len(subTypes),
        aMinVanType[0][0], aPermutationsMap)
    support.tPrint("{} SubType Permutations
        Generated".format(len(subTypePermunations)))

    subSorouPermutations = []

    for permutation in subTypePermunations:
        temp = []
        for index in permutation:
            if index == 0:
                temp.append([False])
            else:
                temp.append(subSorous[index-1])

        newSubSorouPermutations = [p for p in itertools.product(*temp)]
        for item in newSubSorouPermutations:
            if item not in subSorouPermutations:
```

```python
            subSorouPermutations.append(item)

    support.tPrint("{} subSorouPermutations
        Generated".format(len(subSorouPermutations)))

    for i, permutation in enumerate(subSorouPermutations):
        support.tPrint(["New subSorou permutation", '\t', i+1, " of ",
            len(subSorouPermutations)])
        newSorou = genBaseSorou(aMinVanType)
        for index in range(aMinVanType[0][0]):
            if permutation[index]:
                rotationIndicies = findRotationIndicies(permutation[index],
                    newSorou[index][0])

                for rotationIndex in rotationIndicies:
                    workingSubSorou = support.rotate(permutation[index],
                        rotationIndex[0], rotationIndex[1])
                    if support.isSubSorou(workingSubSorou, newSorou[index][0]):
                        temp = support.subtractSorous(newSorou[index][0],
                            workingSubSorou)
                        newSorou[index].append(temp)

        working = selectFromNewSorou(newSorou)
        support.tPrint(["All potential subsorours generated for this permutation"])
        for item in working:
            if len(item) == support.findWeightofType(aMinVanType):
                sortedItem = support.ssort(item)
                itemKey = support.sorouToLatexString(sortedItem)
                if itemKey not in outputKeys:
                    print(item)
                    outputKeys.add(itemKey)
                    output.append(item)

                # if shouldAdd(item, output):
                #     output.append(item)
                #     print(item)

    if output == []:
        print("ERROR: NO SOROUS GENERATED FOR
            {}".format(support.typeToLatexString(aMinVanType)))

    return output

def buildTypeSorouArray (aTypeMatchingArray):
    n = len(aTypeMatchingArray)
    toCheck = set()

    Is = itertools.permutations(range(n))
```

```python
    Js = itertools.permutations(range(n))

    for Iperm in Is:
        for Jperm in Js:
            zipper = zip(Iperm, Jperm)
            toCheck.add(tuple(zipper))

    output = set()

    for item in toCheck:
        shouldAdd = True
        for indexPair in item:
            if not aTypeMatchingArray[indexPair[0]][indexPair[1]]:
                shouldAdd = False

        if shouldAdd:
            output.add(item)

    return output


def makeSubtractedSorouList (aSorou, aSorouList):
    """
    For each sorou in a sorou list, find all ways in which aSorou can be
        subtracted from it
    Returns the list of all possible results from the subtractions
    """
    output = []
    firstRelOrder = support.findRelativeOrder(aSorou)

    for sorou in aSorouList:
        totalRelOrder = support.lcm(firstRelOrder,
            support.findRelativeOrder(sorou))

        for index in range(totalRelOrder):
            rotated = support.rotate(sorou, totalRelOrder, index)
            temp = support.subtractSorous(rotated, aSorou)

            if len(temp) == len(rotated) - len(aSorou) and shouldAdd(temp, output):
                output.append(temp)

    return output


def selectSorous (aSorouListList, aF0):
    """
    Takes [[f_1, f_2, ...]_j, ...] and returns the list [g_1, g_2, ...]
    where each g_i is a sum sorous selected one from each list in aSorouListList
        + aF_0
    """
```

```python
    output = []

    combinations = [p for p in itertools.product(*aSorouListList)]

    for combo in combinations:
        sorou = []
        for item in combo:
            sorou += item

        sorou += aF0
        output.append(sorou)

    return output

def genAllSorousOfType (aType, aF0, aTypeSorouMap, aPermutationsMap):
    if len(aType) == 1:
        return genAllSorousOfMinVanType(aType, aTypeSorouMap, aPermutationsMap)
    else:
        allSubTypeSorous = []
        for aMinVanType in aType:
            allSubTypeSorous.append(genAllSorousOfMinVanType([aMinVanType],
                aTypeSorouMap, aPermutationsMap))

        partitions = partitionF0(aF0, len(aType))

        output = []

        for partition in partitions:
            fullMatchingArray = matchTypeToSorous(partition, allSubTypeSorous)
            typeMatchingArray = simplifyArray(fullMatchingArray)

            subSorouToTypesIndicies = buildTypeSorouArray(typeMatchingArray)

            for matchingIndicies in subSorouToTypesIndicies:

                unselectedSorouLists = []

                for indexPair in matchingIndicies:
                    G_i = partition[indexPair[0]]
                    S_i = aType[indexPair[1]]
                    S_iSorous = allSubTypeSorous[indexPair[1]]

                    subtractedSorouList = makeSubtractedSorouList(G_i, S_iSorous)
                    unselectedSorouLists.append(subtractedSorouList)

                sorouList = selectSorous(unselectedSorouLists, aF0)
```

```python
            for item in sorouList:
                if shouldAdd(item, output):
                    output.append(support.ssort(item))

        return output

def findAllParitiesOfMinVanType (aMinVanType):

    sorous = genAllSorousOfMinVanType(aMinVanType)

    output = set()

    for sorou in sorous:
        par = support.findParity(sorou)
        output.add(par)

    return output

def functionForMap (aType):
    typeStr = support.typeToLatexString(aType)

    if typeStr not in myTypeSorouMap.keys() or myTypeSorouMap[typeStr] == []:
        print(support.findWeightofType(aType), typeStr)

        newSorous = genAllSorousOfMinVanType(aType, myTypeSorouMap,
            myPermutationMap)
        myTypeSorouMap[typeStr] = newSorous
        print()
        with open('finalTypeMap.txt', 'wb') as tm:
            pickle.dump(myTypeSorouMap, tm)
        return newSorous

    else:
        print()
        return myTypeSorouMap[typeStr]


def main():
    global myTypeSorouMap
    global myPermutationMap

    with open("types.txt", "rb") as tls:
        mTypeList = pickle.load(tls)

    with open("finalTypeMap.txt", "rb") as tm:
        myTypeSorouMap = pickle.load(tm)

    myPermutationMap = {}
```

```python
    #### This is the single threaded implementation
    #### Change the iterable here if only specific type's sorous are needed.
    for i,item in enumerate(mTypeList):
        if support.findWeightofType(item) < 22:
            print(i)
            print(item)
            functionForMap(item)


    #### Multithreading the sorou generation is possible using the following:
    # pool = ThreadPool(4) # Set the number of threads here
    # results = pool.map(functionForMap, typesToMap)

    print('All Sorous Generated')

    return 0

if __name__ == '__main__':
    main()
```

B.5. WriteTypeList.py

```python
import support
import pickle

def filterTL (aTypeList):
    output = []
    for item in aTypeList:
        if support.findWeightofType(item) > 21:
            pass
        else:
            output.append(item)
    return output

def hasEquisigned (aPars):
    for item in aPars:
        if item[0] == item[1]:
            return True
    return False

def writeCSV (aTypeList, aTypeMap, aDocName):
    f = open(aDocName, 'w')
    firstLine = 'Weight,\tTop Prime,\tRelative Order,\tWeight
        Partition,\tType,\tHeight,\tParities,\tHasEquisigned\n'
    f.write(firstLine)
```

```python
for item in aTypeList:
    typeKey = support.typeToLatexString(item)
    try:
        tSorous = aTypeMap[typeKey]
    except:
        print("Error for type: " + typeKey)
        continue
    w = support.findWeightofType(item)

    ts = []

    for aSorou in tSorous:
        if support.isMinVan(aSorou): ts.append(aSorou)

    if ts == []: print("ERROR: Type {} has no minimal vanishing sorou in this
        typemap".format(typeKey))

    h = set()
    pars = set()
    for aSorou in ts:
        h.add(support.findHeightofSorou(aSorou))
        newPar = support.findParity(aSorou)
        pars.add(newPar)
    wp = support.findWeightPart(item)

    tp = item[0][0]
    relOrder = support.findRelativeOrder(ts[0])
    partition = support.reverseList(wp)

    partitionStr = '('
    for a in partition:
        partitionStr += str(a) + ';'
    partitionStr = partitionStr[:-1]
    partitionStr += ')'

    parityStr = str()
    for aPar in pars:
        # parityStr += str(aPar) + ';'
        parityStr += '(' + str(aPar[0]) + ';' + str(aPar[1]) + ');'
    parityStr = parityStr[:-1]

    heightStr = str()
    for height in h:
        heightStr += str(height) + ';'
    heightStr = heightStr[:-1]

    line = '{0},\t{1},\t{2},\t{3},\t{4},\t{5},\t{6},\t{7}\n'.format(w,
                                                        tp,
```

```
                                                        relOrder,
                                                        partitionStr,
                                                        support.typeToLatexString(item),
                                                        heightStr,
                                                        parityStr,
                                                        hasEquisigned(pars))

        f.write(line)
        print(line)

    f.close()

def main ():

    with open("types.txt", "rb") as tl:
        typeList = pickle.load(tl)

    with open("finalTypeMap.txt", "rb") as tm:
        typeMap = pickle.load(tm)

    tl = filterTL(typeList)

    print("Writing CSV of type list containing {} types".format(len(tl)))
    writeCSV(tl, typeMap, 'types.csv')
    print("Finsihed writing CSV")

    return 0

if __name__ == '__main__':
    main()
```

L.C., Department of Mathematics, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand

Email address: louis@christie.kiwi.nz

K.D., Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA

Email address: kdykema@math.tamu.edu

I.K., Department of Mathematics, Faculty of Mathematics and Physics, University of Ljubljana, Jadranska 21, 1000 Ljubljana, Slovenia

Email address: igor.klep@fmf.uni-lj.si

Contents