Binary Tree Properties

In this problem, we present a few properties of rooted binary trees. First, let us introduce some notation.

Each node of a tree stores an integer value named key.
Let n be a node in a tree T. We denote by $n.key$ the value of the key stored in n.
By $n.left$ and $n.right$ and $n.parent$ we denote the left and the right child of n and the parent node of n, respectively.
By $n.tree$ we denote the subtree of T rooted in n. The tree $n.tree$ contains all nodes y with the property that n lies on the path from y to the root of T. We denote by $T.root$ the root of the tree T.
By $n.depth$ we denote the depth of node n. The depth of the tree root is 0, for any other node x it holds, $x.depth = x.parent.depth + 1$.

Subsequently, we define eight proprties of nodes and/or (sub)trees.
1. We define the **cost** of a node n to be the value $n.key \times (n.depth+1)$.
2. We define the **disbalance** of node a n to be the absolute value $|SL-SR|$ where SL and SR is the sum of keys of all nodes in $n.left.tree$ and $n.right.tree$, respectively. When a tree is empty we define its sum of keys to be equal to 0.
3. We say that a node n is **2-balanced** if both trees $n.left.tree$ and $n.right.tree$ contain the same number of *2-nodes*. A node is a ***2-node*** if it has exactly 2 children. An empty tree contains 0 *2-nodes*.
4. We say that two distinct nodes x and y are **siblings** if $x.parent = y.parent$. We say that two siblings x and y are **parity siblings** if the parity of $x.key$ is the same as the parity of $y.key$. The parity of an integer z is defined as $z \bmod 2$.
5. We say that a node n is **locally minimal** if $n.key$ is smaller than or equal to the key of any node in $n.tree$. We consider all leaves in the tree to be locally minimal nodes.
6. We say that an internal node n is **weakly dominant** if $n.key$ is bigger than or equal to the value of the key of any leaf in $n.tree$.
7. We say that the tree $n.tree$ is a ***L1-tree*** if it contains at least one node which has only left child and it contains no node which has only right child. Note that a *L1-tree* might be a subtree of another (bigger) *L1-tree*.
8. We say that a sequence of nodes $n_1, n_2, n_3, \ldots, n_k$ is an **increasing path** if it contains at least two nodes and if it also holds
$n_j = n_{j+1}.parent$ and $n_j.key \le n_{j+1}.key$, for $j = 1, 2, \ldots, k-1$.
The **value** of an increasing path is the sum of the keys of all nodes in the increasing path.

Finally, we present a recursive specification of a pseudo-random binary tree.
An integer named structure regulator is associated with each node n and it is denoted by $n.SR$. There is a 9-tuple of non-negative integers (AL, AR, C0, CL, CR, D, M, RK, RSR), where $C0 \le CL \le CR \le M$. The structure and the keys of the left and the right subtree of each node n are defined as follows:

- $T.root.SR$ = RSR.
- If $n.SR$ < C0 or if $n.depth$ = D then node n has no children.
- If C0 ≤ $n.SR$ < CL then node n has only left child and $n.left.SR = (n.SR * AL) \bmod M$.
- If CL ≤ $n.SR$ < CR then node n has only right child and $n.right.SR = (n.SR * AR) \bmod M$.
- If CR ≤ $n.SR$ < M then node n has both left and right children and $n.left.SR = (n.SR * AL) \bmod M$ and $n.right.SR = (n.SR * AR) \bmod M$.

- $T.root.key$ = RK.
- If $n.left$ exists then $n.left.key = (AL * (n.key+1)) \bmod M$.
- If $n.right$ exists then $n.right.key = (AR * (n.key+2)) \bmod M$.

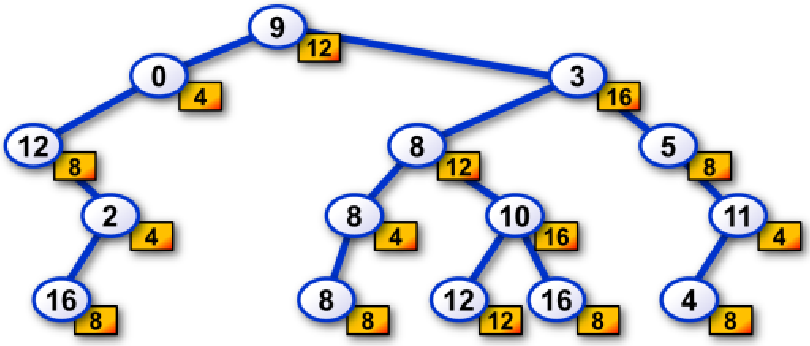

Image 1. A binary tree specified by the nine-tuple (AL, AR, C0, CL, CR, D, M, RK, RSR) = (12, 13, 2, 8, 12, 4, 20, 9, 12). Node keys are drawn in ovals, node structure regulators are drawn in rectangles.
For example, the value of the root structure regulator RSR is 12. It holds, CR = 12 ≤ 12 < 20 = M, therefore the root has two children.
The value of the structure regulator of the left child of the root is (RSR * AL) mod M = (12 * 12) mod 20 = 144 mod 20 = 4.
It holds, C0 = 2 ≤ 4 < 8 = CL, therefore the left child of the root has itself only the left child (grandchild of the root).

The task

You are given a binary tree T. Calculate the following eight characteristics of T:
1. The sum of costs of all nodes in T.
2. The sum of disbalances of all nodes in T.

3. The sum of keys in all 2-balanced nodes in T.
4. The number of parity siblings in T.
5. Sum of keys of all locally minimal nodes in T.
6. The number of weakly dominant nodes in T.
7. The number of *L1-trees* in T.
8. The maximum value of an increasing path in T.

Input

Input consists of one text line with nine integers AL, AR, C0, CL, CR, D, M, RK, RSR, separated by spaces.
It holds 0 ≤ AL, AR, C0, CL, CR, D, M, RK, RSR ≤1000. C0 ≤ CL ≤ CR ≤ M.
It is guaranteed that the tree which is specified by the tuple (AL, AR, C0, CL, CR, D, M, RK, RS) has as at most 100 000 nodes.

Output

Output consists of eight text lines, each line contains a single integer equal to the value of the corresponding property (1. – 8. specified in The task paragraph) of the binary tree which is built according to the input specification.

Example 1

Input

```
12 13 2 8 12 4 20 9 12
```

Output

```
494
214
104
2
87
3
4
37
```

```
      __ 9______________
______ 0      ________ 3__
12____      __ 8____     5____
  __ 2    __ 8  __10__    __11
  16       8    12  16     4
------------------------------
      __12______________
 _____ 4      ________16__
 8____      __12____     8____
  __ 4    __ 4  __16__    __ 4
   8       8    12   8     8
```

Scheme 1. The tree in Example 1. The upper part depicts the keys of the tree, the lower part depicts the structure regulator values associated with each particular node.

Example 2

Input

```
11 12 7 15 23 5 31 28 26
```

Output

```
452
259
66
1
58
1
0
35
```

```
              __28__
______________ 9  19
17________
    ____11__
  __ 8__   1__
   6  27     5
--------------------
              __26__
______________ 7   2
15________
    ____25__
  __27__  21__
  18  14     4
```

Scheme 2. The tree in Example 2. The upper part depicts the keys of the tree, the lower part depicts the structure regulator values associated with each particular node.

Example 3

Input

```
12 13 3 10 16 6 29 18 16
```

Output

```
1645
866
224
1
174
4
7
110
```

```
                          ____18____
__________________________25__  __28
22____________               3   0
      ______22______
    __15____    __22____
  __18  __18  __15  __22__
  25    25    18    15  22
------------------------------------
                          ____16____
__________________________18__  __ 5
13____________               2   2
      ______24______
    __27____    __22____
  __ 5  __ 3  __ 3  __25__
   2     7     7    10   6
```

Scheme 3. The tree in Example 3. The upper part depicts the keys of the tree, the lower part depicts the structure regulator values associated with each particular node.