Programming Assignment 1 (Due Date: 6 March 2020)

For this programming assignment, you will not have to submit any written answers. Instead, you will have to submit a program written in Java or C++11 on CodeCrunch at `https://codecrunch.comp.nus.edu.sg/`. The portal will stop accepting submissions after 6 March 2020 2359h so please start your assignment **early**.

Templates will be provided for all the problems. These templates provide a starting point for your implementation. It is highly recommended to use all the templates given to you. **Do not change the file name or the class name of the template, or else your code may be marked as incorrect**. However, you have to submit your own work. **Posting the question or solution in public repositories are not allowed also counts as a form of plagiarism.** Any form of plagiarism is subject to disciplinary action.

Please include a brief description of your algorithm as a comment at the top of the code as this will be used for marking. You should also explain the time complexity of your algorithm in this comment.

There are some example test cases provided in the assignment folder and these will also be uploaded onto CodeCrunch. If your program fails any of the example test cases, you will get **ZERO** marks. If your program passes all the example test cases, your code will be marked manually so please code neatly and add comments where appropriate. You may be asked to explain your code if the marker cannot understand it. Marks will be deducted if there are bugs in your code or if your algorithm does not meet the time complexity stated in the question. You are strongly encouraged to design your own test cases to test your code.

Note: Passing all the test cases on CodeCrunch does not guarantee that you will get full credit for the assignment.

Background

You are strongly encouraged to read and understand the following articles before attempting the assignment. You are allowed to reuse any part of the code in these 2 articles and can quote them when explaining your algorithm.

- https://www.geeksforgeeks.org/counting-inversions/
- https://www.geeksforgeeks.org/significant-inversions-in-an-array/

Task A (3%)

Given an array $arr[]$ containing N distinct integers, two elements $arr[i]$ and $arr[j]$ form an inversion if $i < j$ and $arr[i] > arr[j]$. We call the inversion **special** if $arr[i] + arr[j]$ is a multiple of P. Write a program to count the number of **special** inversions in the array.

Input

The first line of input contains two integers representing N and P.
The second line of input contains N integers representing the array $arr[]$.

Output

Your program should output the number of **special** inversions on a single line.

Example Input

```
6 4
6 9 2 10 3 2
```

Example Output

```
4
```

Explanation

The special inversions are $(arr[0], arr[2]), (arr[0], arr[5]), (arr[1], arr[4]), (arr[3], arr[5])$.

Limits

- $1 \leq N \leq 2^{16}, 1 \leq P \leq 16N, 1 \leq arr[i] < 2^{30}$
- Your program should terminate within 1 second for C++11 and 2 seconds for Java
- Your algorithm should have a **worst case time complexity** of $O(N \log N)$. Note that the **worst case time complexity** of a hashmap with N elements is $O(N)$ for insertion and deletion.

Task B (4%)

You have N boxes in a row and you want to sort them in non-decreasing order of weight from left to right. You also have N choices of small robots that you can use to sort the boxes and 1 battery containing B charges that can fit into any robot.

The i-th robot has a weight of W_i and a cost of C_i. As the robots are small, they are only able to **swap 2 neighbouring boxes at one time**. Furthermore, if the 2 boxes it swaps have the same weight, or the heavier box is **at most W_i times as heavy as** the lighter box, then it takes 1 charge of the battery to swap them. Otherwise, it takes 2 charges as it needs to stabilise itself from the unbalanced weight. Moving along the row of boxes does not use up battery charges.

You want to choose 1 robot with the least cost that is able to sort the boxes using at most B charges. Find the cost of this robot. It is guaranteed that at least one of the robots can sort the boxes using at most B charges.

Input

The first line of input contains two integers representing N and B.
The second line of input contains N integers representing the weight of the boxes from left to right.
The third line of input contains N integers representing $W_1, W_2, ..., W_N$.
The fourth line of input contains N integers representing $C_1, C_2, ..., C_N$.

Output

Your program should output the least cost on a single line.

Example Input

```
4 7
5 10 5 2
3 1 3 2
6 4 10 8
```

Example Output

```
6
```

Explanation

The second robot requires 8 charges to sort the boxes and thus cannot be used since $B = 7$. The first and third robot only need 5 charges while the fourth robot only needs 7 charges and so all can be used. Among these robots, the first robot has the least cost of 6.

Limits

- $1 \leq N \leq 2^{16}, 1 \leq B, W_i, C_i,$ weight of each box $< 2^{31}$
- Your program should terminate within 2 seconds for C++11 and 4 seconds for Java
- Your algorithm should have a **worst case time complexity** of $O(N \log^2 N)$ which should be independent on the values of B, W_i, C_i and the box weights