Problem G

Greedy Knapsack

Budi stumbled upon the classical 0-1 Knapsack Problem that he has learned from his class in university. There are N items numbered from 1 to N. The i^{th} item has a positive weight of W_i and a positive value of V_i. The objective is to take zero or more items such that their total weight does not exceed M while their total value is maximum.

It has been a while since the last time Budi works on this problem, and he couldn't remember how to solve it. So, Budi devises a greedy algorithm in an attempt to solve this problem. The algorithm goes like this. Each item is processed one by one from the 1^{st} item to the N^{th} item in sequential order. If the i^{th} item can be taken such that the total weight does not exceed M, then you should take that item; otherwise, ignore it. Output the total value of all taken items.

The greedy algorithm can be presented with the following pseudocode.

```
GreedyKnapsack(W[1..N], V[1..N], M):
   total_value = 0
   total_weight = 0
   for i = 1 to N:
      if total_weight + W[i] <= M:
         total_weight = total_weight + W[i]
         total_value = total_value + V[i]
   return total_value
```

Of course, Budi realized that such a greedy algorithm might not produce the optimum solution, but at this point, he doesn't care. Budi noticed that the output of such a greedy algorithm is sensitive to M. So, he decides to run the greedy algorithm with various M ranging from 1 to a given upper limit T and determines what is the largest output that he can get.

Your task in this problem is to help Budi to determine the largest output that he can get from the greedy algorithm by varying the input M from 1 to T.

For example, let $N = 5$, $T = 10$, $W_{1..5} = \{10, 1, 2, 3, 4\}$, and $V_{1.5} = \{1, 1, 1, 1, 1\}$. In this example, the largest output that can be obtained is 3, e.g., with $M = 6$; the greedy algorithm will take the 2^{nd}, 3^{rd}, and 4^{th} items with a total weight of $1 + 2 + 3 = 6$ and a total value of $1 + 1 + 1 = 3$. Notice that if we set $M = 10$, then the greedy algorithm will take only the 1^{st} item with a total weight of 10 and a total value of 1.

Input

Input begins with a line containing two integers N T ($1 \leq N \leq 100\,000$; $1 \leq T \leq 10^{10}$) representing the number of items and the upper limit, respectively. The second line contains N integers W_i ($1 \leq W_i \leq 100\,000$) representing the weight of the i^{th} item. The third line contains N integers V_i ($1 \leq V_i \leq 100\,000$) representing the value of the i^{th} item.

Output

Output contains an integer in a line representing the largest output that can be obtained by the presented greedy algorithm.

Sample Input #1

```
5 10
10 1 2 3 4
1 1 1 1 1
```

Sample Output #1

```
3
```

Explanation for the sample input/output #1

This is the example from the problem statement.

Sample Input #2

```
5 10000000000
10 1 2 3 4
30 2 15 7 11
```

Sample Output #2

```
65
```

Explanation for the sample input/output #2

You can set M to be large enough such that all items can be taken, i.e. $M \geq 20$.

Sample Input #3

```
5 20
4 9 5 1 3
203 175 131 218 304
```

Sample Output #3

```
900
```

Explanation for the sample input/output #3

The largest output can be obtained with $M = 17$. The 1^{st}, 2^{nd}, 4^{th}, and 5^{th} will be taken with a total weight of $4 + 9 + 1 + 3 = 17$ and a total value of $203 + 175 + 218 + 304 = 900$.