Ways to Add

A very simple problem: given a integer N, how many ways can K integers less than N add up to N?

For example, for N = 6 and K = 2, there is 7 ways:

```
0 + 6
1 + 5
2 + 4
3 + 3
4 + 2
5 + 1
6 + 0
```

For N = 3 and K = 3, there is 10 ways.

```
0 + 0 + 3
0 + 1 + 2
0 + 2 + 1
0 + 3 + 0
1 + 0 + 2
1 + 1 + 1
1 + 2 + 0
2 + 0 + 1
2 + 1 + 0
3 + 0 + 0
```

The result could be very large, **please output the answer modulo 1000000007 (10^9 + 7)**.

Input One line containing two integers N, K, 1 <= N,K <= 100.

Output A single line containing the answer modulo 10^9+7.

Example 1

```
Input:
6 2

Output:
7
```

Example 2

```
Input:
3 3

Output:
10
```

Batch Processing

Since Light Kingdom did not pay you salary for a whole year, you decided to leave and work for Conflict Empire. Now, you are again an operator of a super computing center and in control of M nodes.

One day, a research institute from Conflict Empire submitted N computational tasks numbered from 1 to N. Given the running time needed for each task, you are to distribute the tasks among the available nodes such that the node with heaviest workload completes as early as possible. Restriction: every node must process at least one task and must process a contiguous subsection of tasks. That is, you need to find a sequence $0 = L_0 < L_1 < \ldots < L_{M-1} < L_M = N$ where the i-th node processes tasks $L_{i-1}+1, L_{i-1}+2, \ldots, L_i$.

Input

The first line of the input contains two integers N and M (1 <= M <= N <= 500), representing the number of tasks and the number of nodes you control, respectively. The second line contains N integers T_i ($1 \le T_i < 10{,}000{,}000$), representing the time needed to complete each task.

Output

Print one line containing the input $T_1, T_2, \ldots, T_N$ divided into M parts such that the maximum sum of a single part is minimized. You should use character '/' to separate the parts and there must be a space character between every numbers or '/'s.

If there is more than one solution, print the one that minimizes the sum of the first part, then the second part and so on.

Example 1

```
Input:
9 3
100 200 300 400 500 600 700 800 900

Output:
100 200 300 400 500 / 600 700 / 800 900
```

Example 2

```
Input:
5 4
100 100 100 100 100

Output:
100 / 100 / 100 / 100 100
```

Date Overflow

The Year 2000 problem, also known as the Y2K problem, the Millennium bug, Y2K bug, the Y2K glitch, or Y2K, refers to events related to the formatting and storage of calendar data for dates beginning in the year 2000. Problems were anticipated, and arose, because many programs represented four-digit years with only the final two digits – making the year 2000 indistinguishable from 1900. The assumption of a twentieth-century date in such programs could cause various errors, such as the incorrect display of dates and the inaccurate ordering of automated dated records or real-time events.

In fact, there are also many other, similar problems. Suppose you have two computers C1 and C2 with different bugs:

One with the ordinary Y2K bug (i.e. displaying a1:=1900 instead of b1:=2000), and one displaying a2:=1904 instead of b2:=2040 (i.e., after the year 2040, the computer starts showing wrong year, starting with 1904).

Now imagine you see C1 displays the year y1:=1941 and C2 the year y2:=2005.

- You know the actual year cannot be 1941 since in that case C2 will display the same year, and it can not be 2005, either. If the year is 2005, y1 would be 1905, hence it’s impossible.
- Looking at y1, we know the actual year could be 1941, 2041, 2141, etc.
- Then we calculate what C2 would display for 1941, 2041, 2141. It turns it will be 1941, 1905, 2005.
- So it’s possible the actual year is 2141.

To calculate the actual year from those buggy computers requires a lot of work. You are hired to write a program to solve this problem: Find the first possible real year, knowing what some other computers say (yi) and knowing their bugs (displaying ai instead of bi).

Note that the year ai is definitely not after the year the computer was built. Since the actual year can’t be before the year the computers were built, the year your program is looking for can’t be before any ai .

Input

The first line contains 1 integer n, the number of computers, 1 <= n <= 20.

In the following n lines, each line contains 3 integer yi, ai, bi for each computer, 0< = ai <= yi < bi < 10000.

yi is the year the i-th computer displays, bi is the year the bug will happen, and ai is the year displayed instead of bi.

Output

Output an integer z where z is the smallest possible year followed by a newline.

If there is no such year less than 10000, output -1.

Example 1

```
Input:
2
1941 1900 2000
2005 1904 2040

Output:
2141
```

Example 2

```
Input:
2
1941 1900 2000
1940 1900 2000

Output:
-1
```

Example 3

```
3
101 100 200
101 100 201
101 100 202


Output:
101
```

Example 4

```
2
1240 1234 1245
1913 1900 2001


Output:
2923
```

Ayu’s Candies

Ayu has three flavors of candies: banana flavored, grape flavored and cherry flavored. She also has three boxes numbered 1, 2 and 3, each of which contains some of those candies. Now Ayu wants to move candies such that after the movement, each box contains the candies of only one flavor and no two boxes contain same flavored candies. Your task is to help Ayu find the minimum number of candy movements.

Input

The input consists of only one line, containing 9 integers. The first three integers represent the number of banana flavored, grape flavored and cherry flavored candies in the first box, the second three integers represent the number of banana flavored, grape flavored and cherry flavored candies in the second box, and the last three integers represent the number of banana flavored, grape flavored and cherry flavored candies in the third box. It is guaranteed that the total number of candies never exceeds 2^{31}.

For example, the input `10 15 20 30 12 8 15 8 31` indicates that there are 10 banana flavored candies in box 1, 12 grape flavored candies in box 2 and 31 cherry flavored candies in box 3.

Output

Print one line `S x` indicating which flavor of candies go in which box to minimize the number of candy movements. `S` is a string of three letters `G`, `B` and `C` (representing the flavor of grape, banana and cherry, respectively) indicating the flavor associated with each box. `x` is the minimum number of candy movements.

If there is more than one solution, print the alphabetically smallest string as `S`.

Example 1

```
Input:
1 2 3 4 5 6 7 8 9

Output:
BCG 30
```

Example 2

```
Input:
5 10 5 20 10 5 10 20 10

Output:
CBG 50
```

Grocery Delivery

Nia hates shopping and she really despises grocery shopping. She orders her groceries from a local store. In preparation for drone deliveries, the store has new rules about packing items for delivery:

- they will deliver exactly two boxes to the customer (even if the order is small and one of the boxes is empty)
- the capacity of each box is the same and equal to W.

Nia knows the volume occupied by each item on her shopping list: the i^{th}item has the volume v_i. With the new rules, she has to prioritize the items that she orders since not everything can fit in the two boxes. Her list is arranged in the order of priority: from the most important item to the least important one (i.e., the i^{th} item is always more important than the $i+1^{st}$ item). As she prepares the list to send to the store, she only orders the highest priority items and as soon as she comes to an item that is too large to fit in the remaining space in the boxes, she skips that item and all the items below.

The question is how many items from her list she will be able to order and which item should be placed in which box.

Input

The first line contains 2 integers N and W, indicating the number of items and the capacity of a box, 1 <= N <= 200, 1 <= W <= 10000.

The second line contains N integers, indicating the volume of items in Nia's list. They are in the order of priority as described above.

Output

The first line of output should specify the number of items that can be ordered.

For each item that can be ordered, output a line containing "1st" if the item is to be placed in the 1st box and "2nd" if the item is to be placed in the 2nd box. If several arrangements of the items are possible, any one will do.

Example 1

```
Input:
5 10
1 2 100 3 4

Output:
2
1st
1st
```

Explanation of example 1:

Nia has to remove the 3rd item since it’s too large (this means that the 4th and the 5th items are also removed from the order, even though they could fit in the second box).
The first 2 items will be ordered. Any arrangement of the items in boxes will do.

Example 2

```
Input:
7 50
25 30 10 10 15 7 8

Output:
6
1st
2nd
2nd
2nd
1st
1st
```