Educational Codeforces Round 30

A. Chores

2 seconds, 256 megabytes

Luba has to do n chores today. i-th chore takes a_i units of time to complete. It is guaranteed that for every $i \in [2..n]$ the condition $a_i \geq a_{i-1}$ is met, so the sequence is sorted.

Also Luba can work really hard on some chores. She can choose not more than k any chores and do each of them in x units of time instead of a_i ($x < \min_{i=1}^{n} a_i$).

Luba is very responsible, so she has to do all n chores, and now she wants to know the minimum time she needs to do everything. Luba cannot do two chores simultaneously.

Input

The first line contains three integers n, k, x ($1 \leq k \leq n \leq 100$, $1 \leq x \leq 99$) — the number of chores Luba has to do, the number of chores she can do in x units of time, and the number x itself.

The second line contains n integer numbers a_i ($2 \leq a_i \leq 100$) — the time Luba has to spend to do i-th chore.

It is guaranteed that $x < \min_{i=1}^{n} a_i$, and for each $i \in [2..n]$ $a_i \geq a_{i-1}$.

Output

Print one number — minimum time Luba needs to do all n chores.

input
```
4 2 2
3 6 7 10
```
output
```
13
```

input
```
5 2 1
100 100 100 100 100
```
output
```
302
```

In the first example the best option would be to do the third and the fourth chore, spending $x = 2$ time on each instead of a_3 and a_4, respectively. Then the answer is $3 + 6 + 2 + 2 = 13$.

In the second example Luba can choose any two chores to spend x time on them instead of a_i. So the answer is $100 \cdot 3 + 2 \cdot 1 = 302$.

B. Balanced Substring

1 second, 256 megabytes

You are given a string s consisting only of characters `0` and `1`. A substring $[l, r]$ of s is a string $s_l s_{l+1} s_{l+2} \dots s_r$, and its length equals to $r - l + 1$. A substring is called *balanced* if the number of zeroes (`0`) equals to the number of ones in this substring.

You have to determine the length of the longest *balanced* substring of s.

Input

The first line contains n ($1 \leq n \leq 100000$) — the number of characters in s.

The second line contains a string s consisting of exactly n characters. Only characters `0` and `1` can appear in s.

Output

If there is no non-empty *balanced* substring in s, print `0`. Otherwise, print the length of the longest *balanced* substring.

input
```
8
11010111
```
output
```
4
```

input

```
3
111
```

output

```
0
```

In the first example you can choose the substring $[3, 6]$. It is *balanced*, and its length is 4. Choosing the substring $[2, 5]$ is also possible.

In the second example it's impossible to find a non-empty *balanced* substring.

C. Strange Game On Matrix

1 second, 256 megabytes

Ivan is playing a strange game.

He has a matrix a with n rows and m columns. Each element of the matrix is equal to either 0 or 1. Rows and columns are 1-indexed. Ivan can replace any number of ones in this matrix with zeroes. After that, his score in the game will be calculated as follows:

1. Initially Ivan's score is 0;
2. In each column, Ivan will find the topmost 1 (that is, if the current column is j, then he will find minimum i such that $a_{i,j} = 1$). If there are no 1's in the column, this column is skipped;
3. Ivan will look at the next $min(k, n - i + 1)$ elements in this column (starting from the element he found) and count the number of 1's among these elements. This number will be added to his score.

Of course, Ivan wants to maximize his score in this strange game. Also he doesn't want to change many elements, so he will replace the minimum possible number of ones with zeroes. Help him to determine the maximum possible score he can get and the minimum possible number of replacements required to achieve that score.

Input

The first line contains three integer numbers n, m and k ($1 \le k \le n \le 100$, $1 \le m \le 100$).

Then n lines follow, i-th of them contains m integer numbers — the elements of i-th row of matrix a. Each number is either 0 or 1.

Output

Print two numbers: the maximum possible score Ivan can get and the minimum number of replacements required to get this score.

input

```
4 3 2
0 1 0
1 0 1
0 1 0
1 1 1
```

output

```
4 1
```

input

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

output

```
2 0
```

In the first example Ivan will replace the element $a_{1,2}$.

D. Merge Sort

2 seconds, 256 megabytes

Merge sort is a well-known sorting algorithm. The main function that sorts the elements of array a with indices from $[l, r)$ can be implemented as follows:

1. If the segment $[l, r)$ is already sorted in non-descending order (that is, for any i such that $l \le i < r - 1$ $a[i] \le a[i + 1]$), then end the function call;
2. Let $mid = \lfloor \frac{l+r}{2} \rfloor$;
3. Call $mergesort(a, l, mid)$;
4. Call $mergesort(a, mid, r)$;
5. Merge segments $[l, mid)$ and $[mid, r)$, making the segment $[l, r)$ sorted in non-descending order. The merge algorithm doesn't call any other functions.

The array in this problem is 0-indexed, so to sort the whole array, you need to call $mergesort(a, 0, n)$.

The number of calls of function $mergesort$ is very important, so Ivan has decided to calculate it while sorting the array. For example, if $a = \{1, 2, 3, 4\}$, then there will be 1 call of $mergesort$ — $mergesort(0, 4)$, which will check that the array is sorted and then end. If $a = \{2, 1, 3\}$, then the number of calls is 3: first of all, you call $mergesort(0, 3)$, which then sets $mid = 1$ and calls $mergesort(0, 1)$ and $mergesort(1, 3)$, which do not perform any recursive calls because segments $(0, 1)$ and $(1, 3)$ are sorted.

Ivan has implemented the program that counts the number of $mergesort$ calls, but now he needs to test it. To do this, he needs to find an array a such that a is a permutation of size n (that is, the number of elements in a is n, and every integer number from $[1, n]$ can be found in this array), and the number of $mergesort$ calls when sorting the array is exactly k.

Help Ivan to find an array he wants!

Input

The first line contains two numbers n and k ($1 \le n \le 100000$, $1 \le k \le 200000$) — the size of a desired permutation and the number of $mergesort$ calls required to sort it.

Output

If a permutation of size n such that there will be exactly k calls of $mergesort$ while sorting it doesn't exist, output -1. Otherwise output n integer numbers $a[0], a[1], ..., a[n-1]$ — the elements of a permutation that would meet the required conditions. If there are multiple answers, print any of them.

input
```
3 3
```
output
```
2 1 3
```

input
```
4 1
```
output
```
1 2 3 4
```

input
```
5 6
```
output
```
-1
```

E. Awards For Contestants

1 second, 256 megabytes

Alexey recently held a programming contest for students from Berland. n students participated in a contest, i-th of them solved a_i problems. Now he wants to award some contestants. Alexey can award the students with diplomas of three different degrees. Each student either will receive one diploma of some degree, or won't receive any diplomas at all. Let cnt_x be the number of students that are awarded with diplomas of degree x ($1 \le x \le 3$). The following conditions must hold:

- For each x ($1 \le x \le 3$) $cnt_x > 0$;
- For any two degrees x and y $cnt_x \le 2 \cdot cnt_y$.

Of course, there are a lot of ways to distribute the diplomas. Let b_i be the degree of diploma i-th student will receive (or -1 if i-th student won't receive any diplomas). Also for any x such that $1 \le x \le 3$ let c_x be the maximum number of problems solved by a student that receives a diploma of degree x, and d_x be the minimum number of problems solved by a student that receives a diploma of degree x. Alexey wants to distribute the diplomas in such a way that:

1. If student i solved more problems than student j, then he has to be awarded not worse than student j (it's impossible that student j receives a diploma and i doesn't receive any, and also it's impossible that both of them receive a diploma, but $b_j < b_i$);
2. $d_1 - c_2$ is maximum possible;
3. Among all ways that maximize the previous expression, $d_2 - c_3$ is maximum possible;
4. Among all ways that correspond to the two previous conditions, $d_3 - c_{-1}$ is maximum possible, where c_{-1} is the maximum number of problems solved by a student that doesn't receive any diploma (or 0 if each student is awarded with some diploma).

Help Alexey to find a way to award the contestants!

Input

The first line contains one integer number n ($3 \le n \le 3000$).

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 5000$).

Output

Output n numbers. i-th number must be equal to the degree of diploma i-th contestant will receive (or -1 if he doesn't receive any diploma).

If there are multiple optimal solutions, print any of them. It is guaranteed that the answer always exists.

input
```
4
1 2 3 4
```
output
```
3 3 2 1
```

input
```
6
1 4 3 1 1 2
```
output
```
-1 1 2 -1 -1 3
```

F. Forbidden Indices

2 seconds, 256 megabytes

You are given a string s consisting of n lowercase Latin letters. Some indices in this string are marked as *forbidden*.

You want to find a string a such that the value of $|a| \cdot f(a)$ is maximum possible, where $f(a)$ is the number of occurences of a in s such that these occurences end in non-forbidden indices. So, for example, if s is `aaaa`, a is `aa` and index 3 is forbidden, then $f(a) = 2$ because there are three occurences of a in s (starting in indices 1, 2 and 3), but one of them (starting in index 2) ends in a forbidden index.

Calculate the maximum possible value of $|a| \cdot f(a)$ you can get.

Input

The first line contains an integer number n ($1 \le n \le 200000$) — the length of s.

The second line contains a string s, consisting of n lowercase Latin letters.

The third line contains a string t, consisting of n characters `0` and `1`. If i-th character in t is `1`, then i is a forbidden index (otherwise i is not forbidden).

Output

Print the maximum possible value of $|a| \cdot f(a)$.

input
```
5
ababa
00100
```
output
```
5
```

input
```
5
ababa
00000
```
output
```
6
```

input
```
5
ababa
11111
```
output
```
0
```