Partition

Temur and his assistant, Ulug'bek, are preparing a magic trick for the *Uzbekistan Got Talent* show. The trick revolves around Temur solving the following **partition problem**: partition a given collection of positive integers into K non-empty groups such that the sum of the elements in each group is the same. In other words, every integer from the collection must be assigned to exactly one of the K groups, and the sum within each group must be equal. For example, if the given collection is $[2, 1, 6, 4, 5]$ and $K = 3$, a valid partition would be $[2, 4]$, $[1, 5]$, and $[6]$. In this case, the sum of elements in each group is exactly 6.

The trick is performed as follows:

- Ulug'bek and Temur enter separate booths and cannot communicate with each other.
- A jury member gives Ulug'bek an array A of N positive integers, $A[0], A[1], \ldots, A[N-1]$, where each value is between 1 and M inclusive. The jury member also gives Ulug'bek the value of K.
- Ulug'bek chooses at most $K - 1$ (not necessarily distinct) integers to **add as new elements** to the array A. Each added integer must also be between 1 and M inclusive.
- The jury appends the integers chosen by Ulug'bek to the original array. This extended array is then **sorted in non-decreasing order** and is given to Temur, along with the value of K.
- Temur must then solve the partition problem for this extended, sorted array.

Your task is to devise and implement a strategy for Temur and Ulug'bek. It can be proven that, under the given constraints, a strategy for them exists that allows them to successfully solve the partition problem, regardless of the array A provided by the jury.

Implementation Details

You should implement two procedures.

The procedure you should implement for **Ulug'bek** is:

```
std::vector<int> add_numbers(std::vector<int> A, int K, int M)
```

- A: an array of length N representing the original array given to Ulug'bek.
- K: the target number of groups; note that Ulug'bek may add up to $K - 1$ integers to the array.
- M: the maximum allowed value for each original and newly added integer.

- This procedure is called exactly once for each test case.

The procedure should return an array C containing the integers that Ulug'bek wants to add to the original array. Let S denote the length of the array C.

- S must be at most $K - 1$.
- Each element of C must be between 1 and M inclusive.

The procedure you should implement for **Temur** is:

```
std::vector<int> find_partition(std::vector<int> B, int K)
```

- B: an array of length $N + S$ containing both the original integers from A and the integers that Ulug'bek added. The elements of array B are sorted in non-decreasing order.
- K: the target number of groups.
- This procedure is called exactly once for each test case.

The procedure should return an array P that represents the partition of B into K disjoint groups.

- The length of P should be $N + S$.
- For each j such that $0 \leq j < N + S$, $P[j]$ signifies the group to which $B[j]$ belongs.
- The groups should be numbered from 0 to $K - 1$, meaning $0 \leq P[j] < K$ must hold for each $0 \leq j < N + S$.
- For each i between 0 and $K - 1$ inclusive, there must be at least one element j ($0 \leq j < N + S$) such that $P[j] = i$.
- The sum of elements assigned to each of the K groups must be identical.

In the actual grading, a program that calls the above procedures is run **exactly two times**.

- During the first run of the program:
  - `add_numbers` is called exactly once.
  - The returned array is processed by the grading system to compute array B.
- During the second run of the program:
  - `find_partition` is called exactly once.

Constraints

- $3 \leq N \leq 100\,000$
- $2 \leq K \leq 100\,000$
- $K \leq N$
- $1 \leq M \leq 10^9$
- $1 \leq A[i] \leq M$ for each i such that $0 \leq i < N$.

Subtasks

Subtask	Score	Additional Constraints
1	5	$N = 3$
2	4	$M = 1$
3	7	$M \leq 2$
4	10	$N \leq 10$
5	7	$K = 2$
6	12	$K \leq 3$
7	17	$K \leq 10$
8	20	$K \leq 100$
9	18	No additional constraints.

Example

Consider the following call:

```
add_numbers([8, 2, 9, 6, 1, 5, 5], 3, 9)
```

In this example $A = [8, 2, 9, 6, 1, 5, 5]$. We want to partition the numbers into $K = 3$ groups. Ulug'bek can add numbers between 1 and $M = 9$.

The procedure can return $[5, 4]$, meaning that Ulug'bek decides to add $S = 2$ new integers: 5 and 4. These are valid since both elements are between 1 and $M = 9$.

The extended array is then sorted and passed to Temur with the following call:

```
find_partition([1, 2, 4, 5, 5, 5, 6, 8, 9], 3)
```

We might divide the integers from this array into these three groups: $[1, 5, 9]$, $[2, 5, 8]$, and $[4, 5, 6]$. Note that the sum of each of these groups is equal to 15. To report this partition, the procedure should return the array $[0, 1, 2, 0, 1, 2, 2, 1, 0]$.

Sample Grader

Input format:

```
N K M
A[0] A[1] ... A[N-1]
```

Output format:

After the call to `add_numbers` completes, the sample grader:

- Computes the sorted array B.
- Prints arrays C and B in the following format, **followed by an empty line**:

```
S
C[0] C[1] ... C[S-1]
B[0] B[1] ... B[N+S-1]
```

After the call to `find_partition` completes, the grader outputs:

```
L
P[0] P[1] ... P[L-1]
```

Here, L is the length of the array P returned by `find_partition`.