Codeforces β
Sponsored by Telegram

Kotlin Heroes 5: ICPC Round (Practice)

A. A+B (Trial Problem)

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given two integers a and b. Print $a + b$.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case is given as a line of two integers a and b ($-1000 \le a, b \le 1000$).

Output

Print t integers — the required numbers $a + b$.

Example

input

```
4
1 5
314 15
-99 99
123 987
```

output

```
6
329
0
1110
```

B. Candies and Two Sisters

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

There are two sisters Alice and Betty. You have n candies. You want to distribute these n candies between two sisters in such a way that:

- Alice will get a ($a > 0$) candies;
- Betty will get b ($b > 0$) candies;
- each sister will get some **integer** number of candies;
- Alice will get a greater amount of candies than Betty (i.e. $a > b$);
- all the candies will be given to one of two sisters (i.e. $a + b = n$).

Your task is to calculate the number of ways to distribute exactly n candies between sisters in a way described above. Candies are indistinguishable.

Formally, find the number of ways to represent n as the sum of $n = a + b$, where a and b are positive integers and $a > b$.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

The only line of a test case contains one integer n ($1 \le n \le 2 \cdot 10^9$) — the number of candies you have.

Output

For each test case, print the answer — the number of ways to distribute exactly n candies between two sisters in a way described in the problem statement. If there is no way to satisfy all the conditions, print 0.

Example

input

```
6
7
1
2
```

```
3
2000000000
763243547
```

output

```
3
0
0
1
999999999
381621773
```

Note

For the test case of the example, the 3 possible ways to distribute candies are:

- $a = 6$, $b = 1$;
- $a = 5$, $b = 2$;
- $a = 4$, $b = 3$.

C. Equalize Prices Again

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are both a shop keeper and a shop assistant at a small nearby shop. You have n goods, the i-th good costs a_i coins.

You got tired of remembering the price of each product when customers ask for it, thus you decided to simplify your life. More precisely you decided to set the same price for all n goods you have.

However, you don't want to lose any money so you want to choose the price in such a way that the sum of new prices is not less than the sum of the initial prices. It means that if you sell all n goods for the new price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

On the other hand, you don't want to lose customers because of big prices so among all prices you can choose you need to choose the minimum one.

So you need to find the minimum possible equal price of all n goods so if you sell them for this price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

You have to answer q independent queries.

Input

The first line of the input contains one integer q $(1 \le q \le 100)$ — the number of queries. Then q queries follow.

The first line of the query contains one integer n $(1 \le n \le 100)$ — the number of goods. The second line of the query contains n integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^7)$, where a_i is the price of the i-th good.

Output

For each query, print the answer for it — the minimum possible equal price of all n goods so if you sell them for this price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

Example

input

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

output

```
3
2
1
```

D. Construct the String

time limit per test: 4.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given three positive integers n, a and b. You have to construct a string s of length n consisting of lowercase Latin letters such that **each substring** of length a has **exactly** b distinct letters. It is guaranteed that the answer exists.

You have to answer t independent test cases.

Recall that the substring $s[l \dots r]$ is the string $s_l, s_{l+1}, \dots, s_r$ and its length is $r - l + 1$. In this problem you are only interested in substrings of length a.

Input

The first line of the input contains one integer t ($1 \le t \le 2000$) — the number of test cases. Then t test cases follow.

The only line of a test case contains three space-separated integers n, a and b ($1 \le a \le n \le 2000, 1 \le b \le \min(26, a)$), where n is the length of the required string, a is the length of a substring and b is the required number of distinct letters in each substring of length a.

It is guaranteed that the sum of n over all test cases does not exceed 2000 ($\sum n \le 2000$).

Output

For each test case, print the answer — such a string s of length n consisting of lowercase Latin letters that **each substring** of length a has **exactly** b distinct letters. If there are multiple valid answers, print any of them. It is guaranteed that the answer exists.

Example

input

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

output

```
tleelte
qwerty
vvvvvv
abcde
```

Note

In the first test case of the example, consider all the substrings of length 5:

- "`tleel`": it contains 3 distinct (unique) letters,
- "`leelt`": it contains 3 distinct (unique) letters,
- "`eelte`": it contains 3 distinct (unique) letters.

E. Binary String Minimizing

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given a binary string of length n (i. e. a string consisting of n characters '`0`' and '`1`').

In one move you can swap two adjacent characters of the string. What is the lexicographically minimum possible string you can obtain from the given one if you can perform **no more** than k moves? It is possible that you do not perform any moves at all.

Note that you can swap the same pair of adjacent characters with indices i and $i + 1$ arbitrary (possibly, zero) number of times. Each such swap is considered a separate move.

You have to answer q independent test cases.

Input

The first line of the input contains one integer q ($1 \le q \le 10^4$) — the number of test cases.

The first line of the test case contains two integers n and k ($1 \le n \le 10^6, 1 \le k \le n^2$) — the length of the string and the number of moves you can perform.

The second line of the test case contains one string consisting of n characters '`0`' and '`1`'.

It is guaranteed that the sum of n over all test cases does not exceed 10^6 ($\sum n \le 10^6$).

Output

For each test case, print the answer on it: the lexicographically minimum possible string of length n you can obtain from the given one if you can perform **no more** than k moves.

Example

input

```
3
8 5
11011010
7 9
1111100
7 11
1111100
```

output

```
01011110
0101111
0011111
```

Note

In the first example, you can change the string as follows:
$1\underline{10}11010 \rightarrow \underline{10}111010 \rightarrow 0111\underline{10}10 \rightarrow 011\underline{10}110 \rightarrow 01\underline{10}1110 \rightarrow 01011110.$

In the third example, there are enough operations to make the string sorted.

F. Platforms Jumping

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

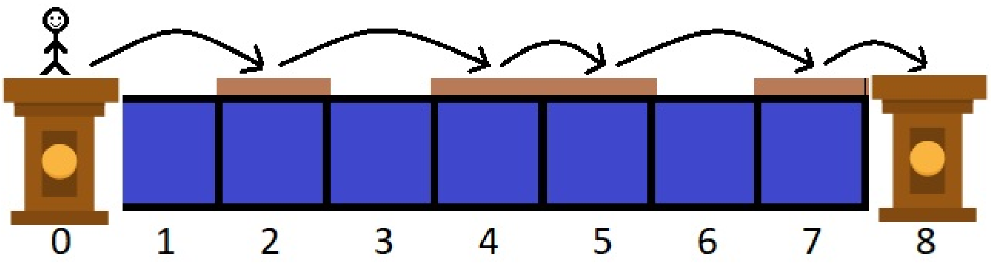
There is a river of width n. The left bank of the river is cell 0 and the right bank is cell $n+1$ (more formally, the river can be represented as a sequence of $n+2$ cells numbered from 0 to $n+1$). There are also m wooden platforms on a river, the i-th platform has length c_i (so the i-th platform takes c_i consecutive cells of the river). It is guaranteed that the sum of lengths of platforms does not exceed n.

You are standing at 0 and want to reach $n+1$ somehow. If you are standing at the position x, you can jump to any position in the range $[x+1; x+d]$. **However** you don't really like the water so you can jump only to such cells that belong to some wooden platform. For example, if $d=1$, you can jump only to the next position (if it belongs to the wooden platform). **You can assume that cells 0 and $n+1$ belong to wooden platforms**.

You want to know if it is possible to reach $n+1$ from 0 if you can move any platform to the left or to the right arbitrary number of times (possibly, zero) as long as they do not intersect each other (but two platforms can touch each other). It also means that you cannot change the relative order of platforms.

Note that you should move platforms until you start jumping (in other words, you first move the platforms and then start jumping).

For example, if $n=7$, $m=3$, $d=2$ and $c=[1,2,1]$, then one of the ways to reach 8 from 0 is follow:



The first example: $n=7$.

Input

The first line of the input contains three integers n, m and d ($1 \le n, m, d \le 1000, m \le n$) — the width of the river, the number of platforms and the maximum distance of your jump, correspondingly.

The second line of the input contains m integers $c_1, c_2, \dots, c_m$ ($1 \le c_i \le n, \sum\limits_{i=1}^{m} c_i \le n$), where c_i is the length of the i-th platform.

Output

If it is impossible to reach $n+1$ from 0, print NO in the first line. Otherwise, print YES in the first line and the array a of length n in the second line — the sequence of river cells (excluding cell 0 and cell $n+1$).

If the cell i does not belong to any platform, a_i should be 0. Otherwise, it should be equal to the index of the platform (1-indexed, platforms are numbered from 1 to m in order of input) to which the cell i belongs.

Note that all a_i equal to 1 should form a contiguous subsegment of the array a of length c_1, all a_i equal to 2 should form a contiguous subsegment of the array a of length c_2, ..., all a_i equal to m should form a contiguous subsegment of the array a of length c_m. The leftmost position of 2 in a should be greater than the rightmost position of 1, the leftmost position of 3 in a should be greater than the rightmost position of 2, ..., the leftmost position of m in a should be greater than the rightmost position of $m-1$.

See example outputs for better understanding.

Examples

input

```
7 3 2
1 2 1
```

output

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

input

```
10 1 11
1
```

output

```
YES
0 0 0 0 0 0 0 0 0 1
```

input

```
10 1 5
2
```

output

```
YES
0 0 0 0 1 1 0 0 0 0
```

Note

Consider the first example: the answer is $[0, 1, 0, 2, 2, 0, 3]$. The sequence of jumps you perform is $0 \to 2 \to 4 \to 5 \to 7 \to 8$.

Consider the second example: it does not matter how to place the platform because you always can jump from 0 to 11.

Consider the third example: the answer is $[0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$. The sequence of jumps you perform is $0 \to 5 \to 6 \to 11$.