# Codeforces Round #638 (Div. 2)

## A. Phoenix and Balance

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix has $n$ coins with weights $2^1, 2^2, \dots, 2^n$. He knows that $n$ is even.

He wants to split the coins into two piles such that each pile has exactly $\frac{n}{2}$ coins and the difference of weights between the two piles is **minimized**. Formally, let $a$ denote the sum of weights in the first pile, and $b$ denote the sum of weights in the second pile. Help Phoenix minimize $|a-b|$, the absolute value of $a-b$.

### Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains an integer $n$ ($2 \le n \le 30$; $n$ is even) — the number of coins that Phoenix has.

### Output

For each test case, output one integer — the minimum possible difference of weights between the two piles.

### Example

input

```
2
2
4
```

output

```
2
6
```

### Note

In the first test case, Phoenix has two coins with weights $2$ and $4$. No matter how he divides the coins, the difference will be $4-2=2$.

In the second test case, Phoenix has four coins of weight $2$, $4$, $8$, and $16$. It is optimal for Phoenix to place coins with weights $2$ and $16$ in one pile, and coins with weights $4$ and $8$ in another pile. The difference is $(2+16)-(4+8)=6$.

## B. Phoenix and Beauty

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix loves beautiful arrays. An array is beautiful if all its subarrays of length $k$ have the same sum. A subarray of an array is any sequence of consecutive elements.

Phoenix currently has an array $a$ of length $n$. He wants to insert some number of integers, possibly zero, into his array such that it becomes beautiful. The inserted integers must be between $1$ and $n$ inclusive. Integers may be inserted anywhere (even before the first or after the last element), and he is **not trying** to minimize the number of inserted integers.

### Input

The input consists of multiple test cases. The first line contains an integer $t$ ($1 \le t \le 50$) — the number of test cases.

The first line of each test case contains two integers $n$ and $k$ ($1 \le k \le n \le 100$).

The second line of each test case contains $n$ space-separated integers ($1 \le a_i \le n$) — the array that Phoenix currently has. This array may or may not be already beautiful.

### Output

For each test case, if it is impossible to create a beautiful array, print -1. Otherwise, print two lines.

The first line should contain the length of the beautiful array $m$ ($n \le m \le 10^4$). You don't need to minimize $m$.

The second line should contain $m$ space-separated integers ($1 \le b_i \le n$) — a beautiful array that Phoenix can obtain after inserting some, possibly zero, integers into his array $a$. You may print integers that weren't originally in array $a$.

If there are multiple solutions, print any. It's guaranteed that if we can make array $a$ beautiful, we can always make it with resulting length no more than $10^4$.

**Example**

**input**

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

**output**

```
5
1 2 1 2 1
4
1 2 2 1
-1
7
4 3 2 1 4 3 2
```

## Note

In the first test case, we can make array $a$ beautiful by inserting the integer $1$ at index $3$ (in between the two existing $2$s). Now, all subarrays of length $k = 2$ have the same sum $3$. There exists many other possible solutions, for example:

- $2, 1, 2, 1, 2, 1$
- $1, 2, 1, 2, 1, 2$

In the second test case, the array is already beautiful: all subarrays of length $k = 3$ have the same sum $5$.

In the third test case, it can be shown that we cannot insert numbers to make array $a$ beautiful.

In the fourth test case, the array $b$ shown is beautiful and all subarrays of length $k = 4$ have the same sum $10$. There exist other solutions also.

## C. Phoenix and Distribution

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix has a string $s$ consisting of lowercase Latin letters. He wants to distribute all the letters of his string into $k$ **non-empty** strings $a_1, a_2, \dots, a_k$ such that every letter of $s$ goes to exactly one of the strings $a_i$. The strings $a_i$ **do not** need to be substrings of $s$. Phoenix can distribute letters of $s$ and rearrange the letters within each string $a_i$ however he wants.

For example, if $s =$ baba and $k = 2$, Phoenix may distribute the letters of his string in many ways, such as:

- ba and ba
- a and abb
- ab and ab
- aa and bb

But these ways are invalid:

- baa and ba
- b and ba
- baba and empty string ($a_i$ should be non-empty)

Phoenix wants to distribute the letters of his string $s$ into $k$ strings $a_1, a_2, \dots, a_k$ to **minimize** the lexicographically maximum string among them, i. e. minimize $max(a_1, a_2, \dots, a_k)$. Help him find the optimal distribution and print the minimal possible value of $max(a_1, a_2, \dots, a_k)$.

String $x$ is lexicographically less than string $y$ if either $x$ is a prefix of $y$ and $x \neq y$, or there exists an index $i$ ( $1 \le i \le min(|x|, |y|)$) such that $x_i$ < $y_i$ and for every $j$ $(1 \le j < i)$ $x_j = y_j$. Here $|x|$ denotes the length of the string $x$.

## Input

The input consists of multiple test cases. The first line contains an integer $t$ $(1 \le t \le 1000)$ — the number of test cases. Each test case consists of two lines.

The first line of each test case consists of two integers $n$ and $k$ $(1 \le k \le n \le 10^5)$ — the length of string $s$ and the number of non-empty strings, into which Phoenix wants to distribute letters of $s$, respectively.

The second line of each test case contains a string $s$ of length $n$ consisting only of lowercase Latin letters.

It is guaranteed that the sum of $n$ over all test cases is $\le 10^5$.

### Output

Print $t$ answers — one per test case. The $i$-th answer should be the minimal possible value of $max(a_1, a_2, \dots, a_k)$ in the $i$-th test case.

### Example

**input**

```
6
4 2
baba
5 2
baacb
5 3
baacb
5 3
aaaaa
6 4
aaxxzz
7 1
phoenix
```

**output**

```
ab
abbc
b
aa
x
ehinopx
```

### Note

In the first test case, one optimal solution is to distribute baba into ab and ab.

In the second test case, one optimal solution is to distribute baacb into abbc and a.

In the third test case, one optimal solution is to distribute baacb into ac, ab, and b.

In the fourth test case, one optimal solution is to distribute aaaaa into aa, aa, and a.

In the fifth test case, one optimal solution is to distribute aaxxzz into az, az, x, and x.

In the sixth test case, one optimal solution is to distribute phoenix into ehinopx.

## D. Phoenix and Science

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix has decided to become a scientist! He is currently investigating the growth of bacteria.

Initially, on day $1$, there is one bacterium with mass $1$.

Every day, some number of bacteria will split (possibly zero or all). When a bacterium of mass $m$ splits, it becomes two bacteria of mass $\frac{m}{2}$ each. For example, a bacterium of mass $3$ can split into two bacteria of mass $1.5$.

Also, every night, the mass of every bacteria will increase by one.

Phoenix is wondering if it is possible for the total mass of all the bacteria to be exactly $n$. If it is possible, he is interested in the way to obtain that mass using the minimum possible number of nights. Help him become the best scientist!

### Input

The input consists of multiple test cases. The first line contains an integer $t$ $(1 \le t \le 1000)$ — the number of test cases.

The first line of each test case contains an integer $n$ $(2 \le n \le 10^9)$ — the sum of bacteria masses that Phoenix is interested in.

### Output

For each test case, if there is no way for the bacteria to exactly achieve total mass $n$, print -1. Otherwise, print two lines.

The first line should contain an integer $d$ — the minimum number of nights needed.

The next line should contain $d$ integers, with the $i$-th integer representing the number of bacteria that should split on the $i$-th day.

If there are multiple solutions, print any.

### Example

**input**

```
3
9
11
2
```

**output**

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

### Note

In the first test case, the following process results in bacteria with total mass $9$:

- Day $1$: The bacterium with mass $1$ splits. There are now two bacteria with mass $0.5$ each.
- Night $1$: All bacteria's mass increases by one. There are now two bacteria with mass $1.5$.
- Day $2$: None split.
- Night $2$: There are now two bacteria with mass $2.5$.
- Day $3$: Both bacteria split. There are now four bacteria with mass $1.25$.
- Night $3$: There are now four bacteria with mass $2.25$.

The total mass is $2.25+2.25+2.25+2.25=9$. It can be proved that $3$ is the minimum number of nights needed. There are also other ways to obtain total mass 9 in 3 nights.

In the second test case, the following process results in bacteria with total mass $11$:

- Day $1$: The bacterium with mass $1$ splits. There are now two bacteria with mass $0.5$.
- Night $1$: There are now two bacteria with mass $1.5$.
- Day $2$: One bacterium splits. There are now three bacteria with masses $0.75$, $0.75$, and $1.5$.
- Night $2$: There are now three bacteria with masses $1.75$, $1.75$, and $2.5$.
- Day $3$: The bacteria with mass $1.75$ and the bacteria with mass $2.5$ split. There are now five bacteria with masses $0.875$, $0.875$, $1.25$, $1.25$, and $1.75$.
- Night $3$: There are now five bacteria with masses $1.875$, $1.875$, $2.25$, $2.25$, and $2.75$.

The total mass is $1.875+1.875+2.25+2.25+2.75=11$. It can be proved that $3$ is the minimum number of nights needed. There are also other ways to obtain total mass 11 in 3 nights.

In the third test case, the bacterium does not split on day $1$, and then grows to mass $2$ during night $1$.

## E. Phoenix and Berries

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix is picking berries in his backyard. There are $n$ shrubs, and each shrub has $a_i$ red berries and $b_i$ blue berries.

Each basket can contain $k$ berries. But, Phoenix has decided that each basket may only contain berries from the same shrub or berries of the same color (red or blue). In other words, all berries in a basket must be from the same shrub or/and have the same color.

For example, if there are two shrubs with $5$ red and $2$ blue berries in the first shrub and $2$ red and $1$ blue berries in the second shrub then Phoenix can fill $2$ baskets of capacity $4$ completely:

- the first basket will contain $3$ red and $1$ blue berries from the first shrub;
- the second basket will contain the $2$ remaining red berries from the first shrub and $2$ red berries from the second shrub.

Help Phoenix determine the maximum number of baskets he can **fill completely**!

### Input

The first line contains two integers $n$ and $k$ ($1 \le n, k \le 500$) — the number of shrubs and the basket capacity, respectively.

The $i$-th of the next $n$ lines contain two integers $a_i$ and $b_i$ ($0 \le a_i, b_i \le 10^9$) — the number of red and blue berries in the $i$-th shrub, respectively.

### Output

Output one integer — the maximum number of baskets that Phoenix can fill completely.

**Examples**

**input**

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

**output**

```
2
```

**input**

```
1 5
2 3
```

**output**

```
1
```

**input**

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

**output**

```
0
```

**input**

```
1 2
1000000000 1
```

**output**

```
500000000
```

### Note

The first example is described above.

In the second example, Phoenix can fill one basket fully using all the berries from the first (and only) shrub.

In the third example, Phoenix cannot fill any basket completely because there are less than $5$ berries in each shrub, less than $5$ total red berries, and less than $5$ total blue berries.

In the fourth example, Phoenix can put all the red berries into baskets, leaving an extra blue berry behind.

## F. Phoenix and Memory

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix is trying to take a photo of his $n$ friends with labels $1, 2, \dots, n$ who are lined up in a row in a special order. But before he can take the photo, his friends get distracted by a duck and mess up their order.

Now, Phoenix must restore the order but he doesn't remember completely! He only remembers that the $i$-th friend from the left had a label between $a_i$ and $b_i$ inclusive. Does there exist a unique way to order his friends based of his memory?

### Input

The first line contains one integer $n$ ($1 \le n \le 2 \cdot 10^5$) — the number of friends.

The $i$-th of the next $n$ lines contain two integers $a_i$ and $b_i$ ($1 \le a_i \le b_i \le n$) — Phoenix's memory of the $i$-th position from the left.

It is guaranteed that Phoenix's memory is valid so there is at least one valid ordering.

### Output

If Phoenix can reorder his friends in a unique order, print YES followed by $n$ integers — the $i$-th integer should be the label of the $i$-th friend from the left.

Otherwise, print NO. Then, **print any two distinct valid orderings** on the following two lines. If are multiple solutions, print any.

### Examples

**input**

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

**output**

```
YES
4 1 2 3
```

**input**

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

**output**

```
NO
1 3 4 2
1 2 4 3
```