A. Most Unstable Array

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers n and m. You have to construct the array a of length n consisting of **non-negative integers** (i.e. integers greater than or equal to zero) such that the sum of elements of this array is **exactly** m and the value $\sum\limits_{i=1}^{n-1} |a_i - a_{i+1}|$ is the maximum possible. Recall that $|x|$ is the absolute value of x.

In other words, you have to maximize the sum of absolute differences between adjacent (consecutive) elements. For example, if the array $a=[1,3,2,5,5,0]$ then the value above for this array is $|1-3|+|3-2|+|2-5|+|5-5|+|5-0|=2+1+3+0+5=11$. Note that this example **doesn't show the optimal answer** but it shows how the required value for some array is calculated.

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 the test case contains two integers n and m ($1 \le n, m \le 10^9$) — the length of the array and its sum correspondingly.

Output

For each test case, print the answer — the maximum possible value of $\sum\limits_{i=1}^{n-1} |a_i - a_{i+1}|$ for the array a consisting of n non-negative integers with the sum m.

Example

input

```
5
1 100
2 2
5 5
2 1000000000
1000000000 1000000000
```

output

```
0
2
10
1000000000
2000000000
```

Note

In the first test case of the example, the only possible array is $[100]$ and the answer is obviously 0.

In the second test case of the example, one of the possible arrays is $[2,0]$ and the answer is $|2-0|=2$.

In the third test case of the example, one of the possible arrays is $[0,2,0,3,0]$ and the answer is $|0-2|+|2-0|+|0-3|+|3-0|=10$.

B. Two Arrays And Swaps

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two arrays a and b both consisting of n positive (greater than zero) integers. You are also given an integer k.

In one move, you can choose two indices i and j ($1 \le i, j \le n$) and swap a_i and b_j (i.e. a_i becomes b_j and vice versa). Note that i and j can be equal or different (in particular, swap a_2 with b_2 or swap a_3 and b_9 both are acceptable moves).

Your task is to find the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k such moves (swaps).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 200$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($1 \le n \le 30; 0 \le k \le n$) — the number of elements in a and b and the maximum number of moves you can do. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 30$), where a_i is the i-th element of a. The third line of the test case contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 30$), where b_i is the i-th element of b.

Output

For each test case, print the answer — the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k swaps.

Example

input

```
5
2 1
1 2
3 4
5 5
5 5 6 6 5
1 2 5 4 3
5 3
1 2 3 4 5
10 9 10 10 9
4 0
2 2 4 3
2 4 2 3
4 4
1 2 2 1
4 4 5 4
```

output

```
6
27
39
11
17
```

Note

In the first test case of the example, you can swap $a_1 = 1$ and $b_2 = 4$, so $a = [4, 2]$ and $b = [3, 1]$.

In the second test case of the example, you don't need to swap anything.

In the third test case of the example, you can swap $a_1 = 1$ and $b_1 = 10$, $a_3 = 3$ and $b_3 = 10$ and $a_2 = 2$ and $b_4 = 10$, so $a = [10, 10, 10, 4, 5]$ and $b = [1, 9, 3, 2, 9]$.

In the fourth test case of the example, you cannot swap anything.

In the fifth test case of the example, you can swap arrays a and b, so $a = [4, 4, 5, 4]$ and $b = [1, 2, 2, 1]$.

C. Board Moves

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a board of size $n \times n$, where n is **odd** (not divisible by 2). Initially, each cell of the board contains one figure.

In one move, you can select **exactly one figure** presented in some cell and move it to one of the cells **sharing a side or a corner with the current cell**, i.e. from the cell (i, j) you can move the figure to cells:

- $(i-1, j-1)$;
- $(i-1, j)$;
- $(i-1, j+1)$;
- $(i, j-1)$;
- $(i, j+1)$;
- $(i+1, j-1)$;
- $(i+1, j)$;
- $(i+1, j+1)$;

Of course, you **can not** move figures to cells out of the board. It is allowed that after a move there will be several figures in one cell.

Your task is to find the minimum number of moves needed to get **all the figures** into **one** cell (i.e. $n^2 - 1$ cells should contain 0 figures and one cell should contain n^2 figures).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 200$) — the number of test cases. Then t test cases follow.

The only line of the test case contains one integer n ($1 \le n < 5 \cdot 10^5$) — the size of the board. It is guaranteed that n is odd (not divisible by 2).

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$ ($\sum n \le 5 \cdot 10^5$).

Output

For each test case print the answer — the minimum number of moves needed to get **all the figures** into **one** cell.

Example

input

```
3
1
5
499993
```

output

```
0
40
41664916690999888
```

D. Constructing the Array

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a of length n consisting of zeros. You perform n actions with this array: during the i-th action, the following sequence of operations appears:

1. Choose the maximum by length subarray (**continuous subsegment**) consisting **only** of zeros, among all such segments choose the **leftmost** one;
2. Let this segment be $[l; r]$. If $r - l + 1$ is odd (not divisible by 2) then assign (set) $a[\frac{l+r}{2}] := i$ (where i is the number of the current action), otherwise (if $r - l + 1$ is even) assign (set) $a[\frac{l+r-1}{2}] := i$.

Consider the array a of length 5 (initially $a = [0, 0, 0, 0, 0]$). Then it changes as follows:

1. Firstly, we choose the segment $[1; 5]$ and assign $a[3] := 1$, so a becomes $[0, 0, 1, 0, 0]$;
2. then we choose the segment $[1; 2]$ and assign $a[1] := 2$, so a becomes $[2, 0, 1, 0, 0]$;
3. then we choose the segment $[4; 5]$ and assign $a[4] := 3$, so a becomes $[2, 0, 1, 3, 0]$;
4. then we choose the segment $[2; 2]$ and assign $a[2] := 4$, so a becomes $[2, 4, 1, 3, 0]$;
5. and at last we choose the segment $[5; 5]$ and assign $a[5] := 5$, so a becomes $[2, 4, 1, 3, 5]$.

Your task is to find the array a of length n after performing all n actions. **Note that the answer exists and unique**.

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 the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer — the array a of length n after performing n actions described in the problem statement. **Note that the answer exists and unique**.

Example

input

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

output

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

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

E. K-periodic Garland

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a garland consisting of n lamps. States of the lamps are represented by the string s of length n. The i-th character of the string s_i equals '0' if the i-th lamp is turned off or '1' if the i-th lamp is turned on. You are also given a positive integer k.

In one move, you can choose **one lamp** and change its state (i.e. turn it on if it is turned off and vice versa).

The garland is called k-periodic if the distance between **each pair of adjacent turned on lamps** is **exactly** k. Consider the case $k = 3$. Then garlands "00010010", "1001001", "00010" and "0" are good but garlands "00101001", "1000001" and "01001100" are not. Note that **the garland is not cyclic**, i.e. the first turned on lamp is not going after the last turned on lamp and vice versa.

Your task is to find the **minimum** number of moves you need to make to obtain k-periodic garland from the given one.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 25\,000$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($1 \le n \le 10^6; 1 \le k \le n$) — the length of s and the required period. The second line of the test case contains the string s 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 — the **minimum** number of moves you need to make to obtain k-periodic garland from the given one.

Example

input

```
6
9 2
010001010
9 3
111100000
7 4
1111111
10 3
1001110101
1 1
1
1 1
0
```

output

```
1
2
5
4
0
0
```

F. Decreasing Heights

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are playing one famous sandbox game with the three-dimensional world. The map of the world can be represented as a matrix of size $n \times m$, where the height of the cell (i, j) is $a_{i,j}$.

You are in the cell $(1, 1)$ right now and want to get in the cell (n, m). You can move only down (from the cell (i, j) to the cell $(i+1, j)$) or right (from the cell (i, j) to the cell $(i, j+1)$). There is an additional **restriction**: if the height of the current cell is x then you can move only to the cell with height $x+1$.

Before the first move you can perform several operations. During one operation, you can decrease the height of **any** cell by one. I.e. you choose some cell (i, j) and assign (set) $a_{i,j} := a_{i,j} - 1$. Note that you **can** make heights **less than or equal to zero**. Also note that you **can** decrease the height of the cell $(1, 1)$.

Your task is to find the **minimum** number of operations you have to perform to obtain at least one suitable path from the cell $(1, 1)$ to the cell (n, m). It is guaranteed that the answer exists.

You have to answer t independent test cases.

Input

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

The first line of the test case contains two integers n and m ($1 \le n, m \le 100$) — the number of rows and the number of columns in the map of the world. The next n lines contain m integers each, where the j-th integer in the i-th line is $a_{i,j}$ ($1 \le a_{i,j} \le 10^{15}$) — the height of the cell (i, j).

It is guaranteed that the sum of n (as well as the sum of m) over all test cases does not exceed 100 ($\sum n \le 100; \sum m \le 100$).

Output

For each test case, print the answer — the **minimum** number of operations you have to perform to obtain at least one suitable path from the cell $(1, 1)$ to the cell (n, m). It is guaranteed that the answer exists.

Example

input

```
5
3 4
1 2 3 4
5 6 7 8
9 10 11 12
5 5
2 5 4 8 3
9 10 11 5 1
12 8 4 2 5
2 2 5 4 1
6 8 2 4 2
2 2
100 10
10 1
1 2
123456789876543 987654321234567
1 1
42
```

output

```
9
49
111
864197531358023
0
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform