Codeforces Round #684 (Div. 2)

A. Buy the String

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

You are given four integers n, c_0, c_1 and h and a binary string s of length n.

A binary string is a string consisting of characters 0 and 1.

You can change any character of the string s (the string should be still binary after the change). You should pay h coins for each change.

After some changes (possibly zero) you want to buy the string. To buy the string you should buy all its characters. To buy the character 0 you should pay c_0 coins, to buy the character 1 you should pay c_1 coins.

Find the minimum number of coins needed to buy the string.

Input

The first line contains a single integer t ($1 \le t \le 10$) — the number of test cases. Next $2t$ lines contain descriptions of test cases.

The first line of the description of each test case contains four integers n, c_0, c_1, h ($1 \le n, c_0, c_1, h \le 1000$).

The second line of the description of each test case contains the binary string s of length n.

Output

For each test case print a single integer — the minimum number of coins needed to buy the string.

Example

input

```
6
3 1 1 1
100
5 10 100 1
01010
5 10 1 1
11111
5 1 10 1
11111
12 2 1 10
101110110101
2 100 1 10
00
```

output

```
3
52
5
10
16
22
```

Note

In the first test case, you can buy all characters and pay 3 coins, because both characters 0 and 1 costs 1 coin.

In the second test case, you can firstly change 2-nd and 4-th symbols of the string from 1 to 0 and pay 2 coins for that. Your string will be 00000. After that, you can buy the string and pay $5 \cdot 10 = 50$ coins for that. The total number of coins paid will be $2 + 50 = 52$.

B. Sum of Medians

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

A *median* of an array of integers of length n is the number standing on the $\lceil \frac{n}{2} \rceil$ (rounding up) position in the non-decreasing ordering of its elements. Positions are numbered starting with 1. For example, a median of the array $[2, 6, 4, 1, 3, 5]$ is equal to 3. **There exist some other definitions of the median, but in this problem, we will use the described one.**

Given two integers n and k and **non-decreasing** array of nk integers. Divide all numbers into k arrays of size n, such that each number belongs to **exactly** one array.

You want the sum of medians of all k arrays to be the maximum possible. Find this maximum possible sum.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. The next $2t$ lines contain descriptions of test cases.

The first line of the description of each test case contains two integers n, k ($1 \le n, k \le 1000$).

The second line of the description of each test case contains nk integers $a_1, a_2, \ldots, a_{nk}$ ($0 \le a_i \le 10^9$) — given array. It is guaranteed that the array is non-decreasing: $a_1 \le a_2 \le \ldots \le a_{nk}$.

It is guaranteed that the sum of nk for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print a single integer — the maximum possible sum of medians of all k arrays.

Example

input

```
6
2 4
0 24 34 58 62 64 69 78
2 2
27 61 81 91
4 3
2 4 16 18 21 27 36 53 82 91 92 95
3 4
3 11 12 22 33 35 38 67 69 71 94 99
2 1
11 41
3 3
1 1 1 1 1 1 1 1 1
```

output

```
165
108
145
234
11
3
```

Note

The examples of possible divisions into arrays for all test cases of the first test:

Test case 1: $[0, 24], [34, 58], [62, 64], [69, 78]$. The medians are $0, 34, 62, 69$. Their sum is 165.

Test case 2: $[27, 61], [81, 91]$. The medians are $27, 81$. Their sum is 108.

Test case 3: $[2, 91, 92, 95], [4, 36, 53, 82], [16, 18, 21, 27]$. The medians are $91, 36, 18$. Their sum is 145.

Test case 4: $[3, 33, 35], [11, 94, 99], [12, 38, 67], [22, 69, 71]$. The medians are $33, 94, 38, 69$. Their sum is 234.

Test case 5: $[11, 41]$. The median is 11. The sum of the only median is 11.

Test case 6: $[1, 1, 1], [1, 1, 1], [1, 1, 1]$. The medians are $1, 1, 1$. Their sum is 3.

C1. Binary Table (Easy Version)

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

This is the easy version of the problem. The difference between the versions is in the number of possible operations that can be made. You can make hacks if and only if you solved both versions of the problem.

You are given a binary table of size $n \times m$. This table consists of symbols 0 and 1.

You can make such operation: select 3 different cells that belong to one 2×2 square and change the symbols in these cells (change 0 to 1 and 1 to 0).

Your task is to make all symbols in the table equal to 0. You are allowed to make at most $3nm$ operations. **You don't need to minimize the number of operations.**

It can be proved that it is always possible.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains two integers n, m ($2 \le n, m \le 100$).

Each of the next n lines contains a binary string of length m, describing the symbols of the next row of the table.

It is guaranteed that the sum of nm for all test cases does not exceed 20000.

Output

For each test case print the integer k ($0 \le k \le 3nm$) — the number of operations.

In the each of the next k lines print 6 integers $x_1, y_1, x_2, y_2, x_3, y_3$ ($1 \le x_1, x_2, x_3 \le n, 1 \le y_1, y_2, y_3 \le m$) describing the next operation. This operation will be made with three cells (x_1, y_1), (x_2, y_2), (x_3, y_3). These three cells should be different. These three cells should belong into some 2×2 square.

Example

input

```
5
2 2
10
11
3 3
011
101
110
4 4
1111
0110
0110
1111
5 5
01011
11001
00010
11011
10000
2 3
011
101
```

output

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

Note

In the first test case, it is possible to make only one operation with cells $(1, 1)$, $(2, 1)$, $(2, 2)$. After that, all symbols will be equal to 0.

In the second test case:

- operation with cells $(2, 1)$, $(3, 1)$, $(3, 2)$. After it the table will be:

```
011
001
000
```

- operation with cells $(1, 2)$, $(1, 3)$, $(2, 3)$. After it the table will be:

```
000
000
000
```

In the fifth test case:

- operation with cells $(1, 3)$, $(2, 2)$, $(2, 3)$. After it the table will be:

```
010
110
```

- operation with cells $(1, 2)$, $(2, 1)$, $(2, 2)$. After it the table will be:

```
000
000
```

C2. Binary Table (Hard Version)

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

This is the hard version of the problem. The difference between the versions is in the number of possible operations that can be made. You can make hacks if and only if you solved both versions of the problem.

You are given a binary table of size $n \times m$. This table consists of symbols 0 and 1.

You can make such operation: select 3 different cells that belong to one 2×2 square and change the symbols in these cells (change 0 to 1 and 1 to 0).

Your task is to make all symbols in the table equal to 0. You are allowed to make at most nm operations. **You don't need to minimize the number of operations.**

It can be proved, that it is always possible.

Input

The first line contains a single integer t ($1 \leq t \leq 5000$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains two integers n, m ($2 \leq n, m \leq 100$).

Each of the next n lines contains a binary string of length m, describing the symbols of the next row of the table.

It is guaranteed, that the sum of nm for all test cases does not exceed 20000.

Output

For each test case print the integer k ($0 \leq k \leq nm$) — the number of operations.

In the each of the next k lines print 6 integers $x_1, y_1, x_2, y_2, x_3, y_3$ ($1 \leq x_1, x_2, x_3 \leq n, 1 \leq y_1, y_2, y_3 \leq m$) describing the next operation. This operation will be made with three cells (x_1, y_1), (x_2, y_2), (x_3, y_3). These three cells should be different. These three cells should belong to some 2×2 square.

Example

input

```
5
2 2
10
11
3 3
011
101
110
4 4
1111
0110
0110
1111
5 5
01011
11001
00010
11011
10000
2 3
011
101
```

output

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

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

Note

In the first test case, it is possible to make only one operation with cells $(1, 1)$, $(2, 1)$, $(2, 2)$. After that, all symbols will be equal to 0.

In the second test case:

- operation with cells $(2, 1)$, $(3, 1)$, $(3, 2)$. After it the table will be:

```
011
001
000
```

- operation with cells $(1, 2)$, $(1, 3)$, $(2, 3)$. After it the table will be:

```
000
000
000
```

In the fifth test case:

- operation with cells $(1, 3)$, $(2, 2)$, $(2, 3)$. After it the table will be:

```
010
110
```

- operation with cells $(1, 2)$, $(2, 1)$, $(2, 2)$. After it the table will be:

```
000
000
```

D. Graph Subset Problem

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given an undirected graph with n vertices and m edges. Also, you are given an integer k.

Find either a clique of size k or a non-empty subset of vertices such that each vertex of this subset has at least k neighbors in the subset. If there are no such cliques and subsets report about it.

A subset of vertices is called a clique of size k if its size is k and there exists an edge between every two vertices from the subset. A vertex is called a neighbor of the other vertex if there exists an edge between them.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains three integers n, m, k ($1 \le n, m, k \le 10^5$, $k \le n$).

Each of the next m lines contains two integers u, v $(1 \le u, v \le n, u \ne v)$, denoting an edge between vertices u and v.

It is guaranteed that there are no self-loops or multiple edges. It is guaranteed that the sum of n for all test cases and the sum of m for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case:

If you found a subset of vertices such that each vertex of this subset has at least k neighbors in the subset in the first line output 1 and the size of the subset. On the second line output the vertices of the subset in any order.

If you found a clique of size k then in the first line output 2 and in the second line output the vertices of the clique in any order.

If there are no required subsets and cliques print -1.

If there exists multiple possible answers you can print any of them.

Example

input

```
3
5 9 4
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
10 15 3
1 2
2 3
3 4
4 5
5 1
1 7
2 8
3 9
4 10
5 6
7 10
10 8
8 6
6 9
9 7
4 5 4
1 2
2 3
3 4
4 1
1 3
```

output

```
2
4 1 2 3
1 10
1 2 3 4 5 6 7 8 9 10
-1
```

Note

In the first test case: the subset $\{1, 2, 3, 4\}$ is a clique of size 4.

In the second test case: degree of each vertex in the original graph is at least 3. So the set of all vertices is a correct answer.

In the third test case: there are no cliques of size 4 or required subsets, so the answer is -1.

E. Greedy Shopping

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a_1, a_2, \ldots, a_n$ of integers. This array is **non-increasing**.

Let's consider a line with n shops. The shops are numbered with integers from 1 to n from left to right. The cost of a meal in the i-th shop is equal to a_i.

You should process q queries of two types:

- 1 x y: for each shop $1 \leq i \leq x$ set $a_i = max(a_i, y)$.
- 2 x y: let's consider a hungry man with y money. He visits the shops from x-th shop to n-th and if he can buy a meal in the current shop he buys one item of it. Find how many meals he will purchase. The man can buy a meal in the shop i if he has at least a_i money, and after it his money decreases by a_i.

Input

The first line contains two integers n, q ($1 \leq n, q \leq 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$ — the costs of the meals. It is guaranteed, that $a_1 \geq a_2 \geq \ldots \geq a_n$.

Each of the next q lines contains three integers t, x, y ($1 \leq t \leq 2$, $1 \leq x \leq n$, $1 \leq y \leq 10^9$), each describing the next query.

It is guaranteed that there exists at least one query of type 2.

Output

For each query of type 2 output the answer on the new line.

Example

input

```
10 6
10 10 10 6 6 5 5 5 3 1
2 3 50
2 4 10
1 3 10
2 2 36
1 4 7
2 2 17
```

output

```
8
3
6
2
```

Note

In the first query a hungry man will buy meals in all shops from 3 to 10.

In the second query a hungry man will buy meals in shops 4, 9, and 10.

After the third query the array $a_1, a_2, \ldots, a_n$ of costs won't change and will be $\{10, 10, 10, 6, 6, 5, 5, 5, 3, 1\}$.

In the fourth query a hungry man will buy meals in shops 2, 3, 4, 5, 9, and 10.

After the fifth query the array a of costs will be $\{10, 10, 10, 7, 6, 5, 5, 5, 3, 1\}$.

In the sixth query a hungry man will buy meals in shops 2 and 4.