Educational Codeforces Round 43 (Rated for Div. 2)

A. Minimum Binary Number

1 second, 256 megabytes

String can be called *correct* if it consists of characters "0" and "1" and there are no redundant leading zeroes. Here are some examples: "0", "10", "1001".

You are given a *correct* string s.

You can perform two different operations on this string:

1. swap any pair of adjacent characters (for example, "101" "110");
2. replace "11" with "1" (for example, "110" "10").

Let $val(s)$ be such a number that s is its binary representation.

Correct string a is less than some other *correct* string b iff $val(a) < val(b)$.

Your task is to find the minimum *correct* string that you can obtain from the given one using the operations described above. You can use these operations any number of times in any order (or even use no operations at all).

Input

The first line contains integer number n ($1 \le n \le 100$) — the length of string s.

The second line contains the string s consisting of characters "0" and "1". It is guaranteed that the string s is *correct*.

Output

Print one string — the minimum *correct* string that you can obtain from the given one.

input
4 1001
output
100

input
1 1
output
1

In the first example you can obtain the answer by the following sequence of operations: "1001" "1010" "1100" "100".

In the second example you can't obtain smaller answer no matter what operations you use.

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

B. Lara Croft and the New Game

2 seconds, 256 megabytes

You might have heard about the next game in Lara Croft series coming out this year. You also might have watched its trailer. Though you definitely missed the main idea about its plot, so let me lift the veil of secrecy.

Lara is going to explore yet another dangerous dungeon. Game designers decided to use good old 2D environment. The dungeon can be represented as a rectangle matrix of n rows and m columns. Cell (x, y) is the cell in the x-th row in the y-th column. Lara can move between the neighbouring by side cells in all four directions.

Moreover, she has even chosen the path for herself to avoid all the traps. She enters the dungeon in cell $(1, 1)$, that is top left corner of the matrix. Then she goes down all the way to cell $(n, 1)$ — the bottom left corner. Then she starts moving in the snake fashion — all the way to the right, one cell up, then to the left to the cell in 2-nd column, one cell up. She moves until she runs out of non-visited cells. n and m given are such that she always end up in cell $(1, 2)$.

Lara has already moved to a neighbouring cell k times. Can you determine her current position?

Input

The only line contains three integers n, m and k ($2 \le n, m \le 10^9$, **n is always even**, $0 \le k < n \cdot m$). Note that k doesn't fit into 32-bit integer type!

Output

Print the cell (the row and the column where the cell is situated) where Lara ends up after she moves k times.

input
4 3 0
output
1 1

input
4 3 11
output
1 2

input
4 3 7
output
3 2

Here is her path on matrix 4 by 3:

C. Nested Segments

2 seconds, 256 megabytes

You are given a sequence $a_1, a_2, ..., a_n$ of one-dimensional segments numbered 1 through n. Your task is to find two distinct indices i and j such that segment a_i lies within segment a_j.

Segment $[l_1, r_1]$ lies within segment $[l_2, r_2]$ iff $l_1 \geq l_2$ and $r_1 \leq r_2$.

Print indices i and j. If there are multiple answers, print any of them. If no answer exists, print `-1 -1`.

Input

The first line contains one integer n ($1 \leq n \leq 3 \cdot 10^5$) — the number of segments.

Each of the next n lines contains two integers l_i and r_i ($1 \leq l_i \leq r_i \leq 10^9$) — the i-th segment.

Output

Print two distinct indices i and j such that segment a_i lies within segment a_j. If there are multiple answers, print any of them. If no answer exists, print `-1 -1`.

input
5 1 10 2 9 3 9 2 3 2 9
output
2 1

input
3 1 5 2 6 6 20
output
-1 -1

In the first example the following pairs are considered correct:

- $(2, 1), (3, 1), (4, 1), (5, 1)$ — not even touching borders;
- $(3, 2), (4, 2), (3, 5), (4, 5)$ — touch one border;
- $(5, 2), (2, 5)$ — match exactly.

D. Degree Set

2 seconds, 256 megabytes

You are given a sequence of n positive integers $d_1, d_2, ..., d_n$ ($d_1 < d_2 < ... < d_n$). Your task is to construct an undirected graph such that:

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

- there are exactly $d_n + 1$ vertices;
- there are no self-loops;
- there are no multiple edges;
- there are no more than 10^6 edges;
- its *degree set* is equal to d.

Vertices should be numbered 1 through $(d_n + 1)$.

Degree sequence is an array a with length equal to the number of vertices in a graph such that a_i is the number of vertices adjacent to i-th vertex.

Degree set is a sorted in increasing order sequence of all distinct values from the *degree sequence*.

It is guaranteed that there exists such a graph that all the conditions hold, and it contains no more than 10^6 edges.

Print the resulting graph.

Input

The first line contains one integer n ($1 \leq n \leq 300$) — the size of the degree set.

The second line contains n integers $d_1, d_2, ..., d_n$ ($1 \leq d_i \leq 1000$, $d_1 < d_2 < ... < d_n$) — the degree set.

Output

In the first line print one integer m ($1 \leq m \leq 10^6$) — the number of edges in the resulting graph. It is guaranteed that there exists such a graph that all the conditions hold and it contains no more than 10^6 edges.

Each of the next m lines should contain two integers v_i and u_i ($1 \leq v_i, u_i \leq d_n + 1$) — the description of the i-th edge.

input
3 2 3 4
output
8 3 1 4 2 4 5 2 5 5 1 3 2 2 1 5 3

input
3 1 2 3
output
4 1 2 1 3 1 4 2 3

E. Well played!

1 second, 256 megabytes

Recently Max has got himself into popular CCG "BrainStone". As "BrainStone" is a pretty intellectual game, Max has to solve numerous hard problems during the gameplay. Here is one of them:

Max owns n creatures, i-th of them can be described with two numbers — its health hp_i and its damage dmg_i. Max also has two types of spells in stock:

1. Doubles health of the creature ($hp_i := hp_i \cdot 2$);
2. Assigns value of **health** of the creature to its **damage** ($dmg_i := hp_i$).

Spell of first type can be used no more than a times in total, of the second type — no more than b times in total. Spell can be used on a certain creature multiple times. Spells can be used in arbitrary order. It isn't necessary to use all the spells.

Max is really busy preparing for his final exams, so he asks you to determine what is the maximal total damage of all creatures he can achieve if he uses spells in most optimal way.

Input

The first line contains three integers n, a, b ($1 \le n \le 2 \cdot 10^5$, $0 \le a \le 20$, $0 \le b \le 2 \cdot 10^5$) — the number of creatures, spells of the first type and spells of the second type, respectively.

The i-th of the next n lines contain two number hp_i and dmg_i ($1 \le hp_i, dmg_i \le 10^9$) — description of the i-th creature.

Output

Print single integer — maximum total damage creatures can deal.

input
2 1 1 10 15 6 1

output
27

input
3 0 3 10 8 7 11 5 2

output
26

In the first example Max should use the spell of the first type on the second creature, then the spell of the second type on the same creature. Then total damage will be equal to $15 + 6 \cdot 2 = 27$.

In the second example Max should use the spell of the second type on the first creature, then the spell of the second type on the third creature. Total

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js

F. Minimal k-covering

1.5 seconds, 256 megabytes

You are given a bipartite graph $G = (U, V, E)$, U is the set of vertices of the first part, V is the set of vertices of the second part and E is the set of edges. There might be multiple edges.

Let's call some subset of its edges k-*covering* iff the graph has each of its vertices incident to at least k edges. *Minimal* k-*covering* is such a k-covering that the size of the subset is minimal possible.

Your task is to find minimal k-covering for each , where $minDegree$ is the minimal degree of any vertex in graph G.

Input

The first line contains three integers n_1, n_2 and m ($1 \le n_1, n_2 \le 2000$, $0 \le m \le 2000$) — the number of vertices in the first part, the number of vertices in the second part and the number of edges, respectively.

The i-th of the next m lines contain two integers u_i and v_i ($1 \le u_i \le n_1$, $1 \le v_i \le n_2$) — the description of the i-th edge, u_i is the index of the vertex in the first part and v_i is the index of the vertex in the second part.

Output

For each print the subset of edges (minimal k-covering) in separate line.

The first integer cnt_k of the k-th line is the number of edges in minimal k-covering of the graph. Then cnt_k integers follow — original indices of the edges which belong to the minimal k-covering, these indices should be pairwise distinct. Edges are numbered 1 through m in order they are given in the input.

input
3 3 7 1 2 2 3 1 3 3 2 3 3 2 1 2 1

output
0 3 3 7 4 6 1 3 6 7 4 5

input
1 1 5 1 1 1 1 1 1 1 1 1 1

output

```
0
1 5
2 4 5
3 3 4 5
4 2 3 4 5
5 1 2 3 4 5
```

Codeforces (c) Copyright 2010-2020 Mike Mirzayanov
The only programming contests Web 2.0 platform

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js