Evolutionary Computation, 2017/2018
Programming assignment 1

Important information

- Deadline: 11/Oct/2017, 23:59.
- All problems must be submitted through Mooshak.
- Please go to `http://mooshak.deei.fct.ualg.pt/~mooshak/` and register by doing the following steps:
  - Choose the link *Register for online contest*
  - Select 'EC1718' from the 'contest' list box, fill in your name and email, and select the group `students`. You will receive an email with your password and you will be able to submit your programming assignments on Mooshak.
- You can submit programs in C, C++, Java, or Python.

About this assignment

The purpose of this programming assignment is to get you involved in programming basic components of a simple genetic algorithm.

Problem A: Max

This is just a warmup exercise to get you familiarized with Mooshak, the online platform that automatically judges your computer programs.

Given a collection of n integers, your program should output the maximum element of the collection.

Input

The input consists of an integer n that specifies the size of the collection, followed by the n integers themselves, one per line.

Output

The output should be the maximum element from the collection, followed by a newline character.

Example

Sample Input

```
4
2
-3
21
8
```

Sample Output

```
21
```

Problem B: Initializing a population

You are going to initialize a population of N binary strings of length ℓ. As you know, the genetic algorithm uses pseudo random numbers in its operation, and the output is non-deterministic. Mooshak requires deterministic outputs, i.e., for a given input, only a specific output should be considered correct.

In order to eliminate the non-determinism, the random numbers are going to be given to you as part of the input. Concretely, you will be given $N * \ell$ uniformly generated random numbers in $[0..1)$. Let u be one such number. If you want to generate a random bit b, you should assume that $b = 1$ if $u < 0.5$, and $b = 0$ otherwise. You should use the first ℓ bits to generate the first string, the next ℓ bits to generate the second string, and so on.

Input

The input contains $2 + N * \ell$ lines. The first line contains an integer N that specifies the size of the population. The second line contains an integer ℓ that specifies the string (chromosome) length. Each of the next $N * \ell$ lines contain a real number in $[0..1)$ that is supposed to be a pseudo-random number as described previously.

For the purpose if this exercise, you can assume that both N and ℓ are less or equal than 100.

Output

The output should be a population of N binary strings of length ℓ, with each string being followed by a newline character.

Example

Sample Input

```
4
3
0.23432
0.43333
0.91111
0.23556
0.71141
0.80345
0.56244
0.44256
0.99924
0.42672
0.85034
0.37256
```

Sample Output

```
110
100
010
101
```

Problem C: onemax

The onemax problem is probably the most widely tested problem in the evolutionary computation field. The objective of the problem is to maximize a function that returns the number of 1s in an input string.

Input

The input has 2 lines. The first line contains an integer ℓ that specifies the string length. The second line contains the input binary string of length ℓ. You can assume that $\ell \leq 1000$.

Output

The output should be the value returned by the onemax function, followed by a newline character.

Example

Sample Input

```
8
11010011
```

Sample Output

```
5
```

Problem D: $f(x) = x^2$

You should implement another fitness function. This one interprets the input string as the binary representation of an integer in base 10, and returns the square of that number. You should consider the leftmost bit to be the most significant one.

Input

The input has 2 lines. The first line contains an integer ℓ that specifies the string length. The second line contains the input binary string of length ℓ. You can assume that $\ell \leq 10$.

Output

The output should be the value returned by the function, followed by a newline character.

Example

Sample Input

```
4
1101
```

Sample Output

```
169
```

Problem E: binary tournament selection

You should implement tournament selection as explained in class. Given a population of N strings with the corresponding fitness values, you are supposed to simulate N tournaments. For each tournament you should pick 2 solutions at random from the population. The winner goes to the next phase.

In case of a tie, please assume that the first solution is the winner of the tournament. In order to perform this operation on a population of size N, you are going to need $2 * N$ random numbers which will be given to you as part of the input. Once again, each random number will be given in the interval $[0..1)$, and you will need to map it into an integer random number in the $[a..b]$ interval. (Think about what is a and b in your case). To do that you can do the following procedure. Let u be a random number in $[0..1)$. The desired integer random number in $[a..b]$ can be calculated as $a + \lfloor u * (b - a + 1) \rfloor$. (Please think why this procedure is correct.)

Input

The input has $2 + 3N$ lines. The first line contains an integer N that specifies the size of the population. The second line contains an integer ℓ that specifies the string length. Each of the next N lines contain a binary string of length ℓ, followed by a white space, followed by string's fitness value (a real number). Then, each of the next $2 * N$ lines contains a pseudo random number uniformly generated in $[0..1)$.

For the purpose if this exercise, you can assume that both N and ℓ are less or equal than 100.

Output

The output should be the population of selected strings, i.e., those that won the tournaments. You should send them to the output in the same order as the tournaments take place. That is, the first string sent to the output should be winner of the first tournament, the second string sent to the output should be the winner of the second tournament, and so on.

Example

Sample Input

```
4
3
110 8
001 7
101 23
111 5
0.16758
0.54902
0.97234
```

```
0.25456
0.19763
0.83764
0.76089
0.47892
```

Sample Output

```
101
001
110
001
```