10212 The Last Non-zero Digit

In this problem you will be given two decimal integer number N, M. You will have to find the last non-zero digit of the P_M^N. This means no of permutations of N things taking M at a time.

Input

The input file contains several lines of input. Each line of the input file contains two integers N ($0 \leq N \leq 20000000$), M ($0 \leq M \leq N$). Input is terminated by end-of-file.

Output

For each line of the input file you should output a single digit, which is the last non-zero digit of P_M^N. For example, if P_M^N is 720 then the last non-zero digit is 2. So in this case your output should be 2.

Sample Input

```
10 10
10 5
25 6
```

Sample Output

```
8
4
2
```

990 Diving for gold

John is a diver and a treasure hunter. He has just found the location of a pirate ship full of treasures. The sofisticated sonar system on board his ship allows him to identify the location, depth and quantity of gold in each suken treasure. Unfortunatelly, John forgot to bring a GPS device and the chances of ever finding this location again are very slim so he has to grab the gold now. To make the situation worse, John has only one compressed air bottle.

John wants to dive with the compressed air bottle to recover as much gold as possible from the wreck. Write a program John can use to select which treasures he should pick to maximize the quantity of gold recovered.

The problem has the following restrictions:

- There are n treasures $\{(d_1, v_1), (d_2, v_2), \ldots (d_n, v_n)\}$ represented by the pair (depth, quantity of gold). There are at most 30 treasures.
- The air bottle only allows for t seconds under water. t is at most 1000 seconds.
- In each dive, John can bring the maximum of one treasure at a time.
- The descent time is $td_i = w * d_i$, where w is an integer constant.
- The ascent time is $ta_i = 2w * d_i$, where w is an integer constant.
- Due to instrument limitations, all parameters are integer.

Input

The input to this program consists of a sequence of integer values. Input contains several test cases. The first line of each dataset should contain the values t and w. The second line contains the number of treasures. Each of the following lines contains the depth d_i and quantity of gold v_i of a different treasure.

A blank line separates each test case.

Note:

In this sample input, the bottle of compressed air has a capacity of 200 seconds, the constant w has the value 4 and there are 3 treasures, the first one at a depth of 10 meters and worth 5 coins of gold, the second one at a depth of 10 meters and worth 1 coin of gold, and the third one at 7 meters and worth 2 coins of gold.

Output

The first line of the output for each dataset should contain the maximum amount of gold that John can recover from the wreck. The second line should contain the number of recovered treasures. Each of the following lines should contain the depth and amount of gold of each recovered treasure. Treasures should be presented in the same order as the input file.

Print a blank line between outputs for different datasets.

Sample Input

```
210 4
3
10 5
10 1
7 2
```

Sample Output

```
7
2
10 5
7 2
```

11264 Coin Collector

Our dear Sultan is visiting a country where there are n different types of coin. He wants to collect as many different types of coin as you can. Now if he wants to withdraw X amount of money from a Bank, the Bank will give him this money using following algorithm.

```
withdraw(X){
    if( X == 0) return;
    Let Y be the highest valued coin that does not exceed X.
    Give the customer Y valued coin.
    withdraw(X-Y);
}
```

Now Sultan can withdraw any amount of money from the Bank. He should maximize the number of different coins that he can collect in a single withdrawal.

Input

First line of the input contains T the number of test cases. Each of the test cases starts with n ($1 \leq n \leq 1000$), the number of different types of coin. Next line contains n integers C_1, C_2, ..., C_n the value of each coin type. $C_1 < C_2 < C_3 < \ldots < C_n < 1000000000$. C_1 equals to 1.

Output

For each test case output one line denoting the maximum number of coins that Sultan can collect in a single withdrawal. He can withdraw infinite amount of money from the Bank.

Sample Input

```
2
6
1 2 4 8 16 32
6
1 3 6 8 15 20
```

Sample Output

```
6
4
```

Problem B
Squeeze the Cylinders

Input: Standard Input
Time Limit: 1 second

Laid on the flat ground in the stockyard are a number of heavy metal cylinders with (possibly) different diameters but with the same length. Their ends are aligned and their axes are oriented to exactly the same direction.

We'd like to minimize the area occupied. The cylinders are too heavy to lift up, although rolling them is not too difficult. So, we decided to push the cylinders with two high walls from both sides.

Your task is to compute the minimum possible distance between the two walls when cylinders are squeezed as much as possible. Cylinders and walls may touch one another. They cannot be lifted up from the ground, and thus their order cannot be altered.

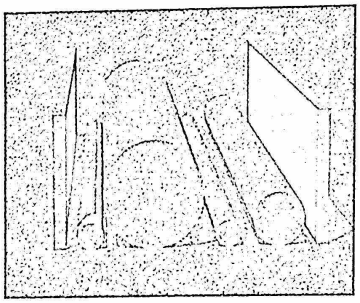

Figure B.1. Cylinders between two walls

Input

The input consists of a single test case. The first line has an integer N ($1 \leq N \leq 500$), which is the number of cylinders. The second line has N positive integers at most 10,000. They are the radii of cylinders from one side to the other.

Output

Print the distance between the two walls when they fully squeeze up the cylinders. The number should not contain an error greater than 0.0001.

Sample Input 1	Sample Output 1
2 10 10	40.00000000

Sample Input 2	Sample Output 2
2 4 12	29.85640646

Sample Input 3	Sample Output 3
5 1 10 1 10 1	40.00000000

Sample Input 4	Sample Output 4
3 1 1 1	6.00000000

Sample Input 5	Sample Output 5
2 5000 10000	29142.13562373

The following figures correspond to the Sample 1, 2, and 3.

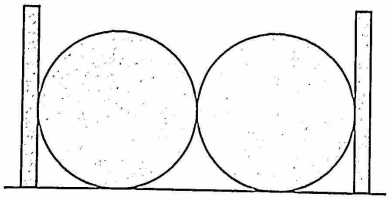

Figure B.2. Sample 1

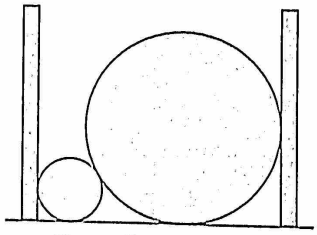

Figure B.3. Sample 2

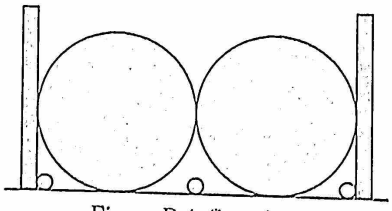

Figure B.4. Sample 3

Problem C
Sibling Rivalry

Input: Standard Input
Time Limit: 2 seconds

You are playing a game with your elder brother.

First, a number of circles and arrows connecting some pairs of the circles are drawn on the ground. Two of the circles are marked as the *start circle* and the *goal circle*.

At the start of the game, you are on the start circle. In each *turn* of the game, your brother tells you a number, and you have to take that number of *steps*. At each step, you choose one of the arrows outgoing from the circle you are on, and move to the circle the arrow is heading to. You can visit the same circle or use the same arrow any number of times.

Your aim is to stop on the goal circle after the fewest possible turns, while your brother's aim is to prevent it as long as possible. Note that, in each single turn, you *must* take the exact number of steps your brother tells you. Even when you visit the goal circle during a turn, you have to leave it if more steps are to be taken.

If you reach a circle with no outgoing arrows before completing all the steps, then you lose the game. You also have to note that, your brother may be able to repeat turns forever, not allowing you to stop after any of them.

Your brother, mean but not too selfish, thought that being allowed to choose arbitrary numbers is not fair. So, he decided to declare three numbers at the start of the game and to use only those numbers.

Your task now is, given the configuration of circles and arrows, and the three numbers declared, to compute the smallest possible number of turns within which you can always finish the game, no matter how your brother chooses the numbers.

Input

The input consists of a single test case, formatted as follows.

$n\ m\ a\ b\ c$
$u_1\ v_1$
$\vdots$
$u_m\ v_m$

All numbers in a test case are integers. n is the number of circles ($2 \leq n \leq 50$). Circles are numbered 1 through n. The start and goal circles are numbered 1 and n, respectively. m is the number of arrows ($0 \leq m \leq n(n-1)$). a, b, and c are the three numbers your brother declared ($1 \leq a, b, c \leq 100$). The pair, u_i and v_i, means that there is an arrow from the circle u_i to the circle v_i. It is ensured that $u_i \neq v_i$ for all i, and $u_i \neq u_j$ or $v_i \neq v_j$ if $i \neq j$.

Output

Print the smallest possible number of turns within which you can always finish the game. Print `IMPOSSIBLE` if your brother can prevent you from reaching the goal, by either making you repeat the turns forever or leading you to a circle without outgoing arrows.

Sample Input 1

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

Sample Output 1

```
IMPOSSIBLE
```

Sample Input 2

```
8 12 1 2 3
1 2
2 3
1 4
2 4
3 4
1 5
5 8
4 6
6 7
4 8
6 8
7 8
```

Sample Output 2

```
2
```

For Sample Input 1, your brother may choose 1 first, then 2, and repeat these forever. Then you can never finish.

For Sample Input 2 (Figure C.1), if your brother chooses 2 or 3, you can finish with a single turn. If he chooses 1, you will have three options.

- Move to the circle 5. This is a bad idea: Your brother may then choose 2 or 3 and make you lose.
- Move to the circle 4. This is the best choice: From the circle 4, no matter any of 1, 2, or 3 your brother chooses in the next turn, you can finish immediately.

- Move to the circle 2. This is not optimal for you. If your brother chooses 1 in the next turn, you cannot finish yet. It will take three or more turns in total.

In summary, no matter how your brother acts, you can finish within two turns. Thus the answer is 2.

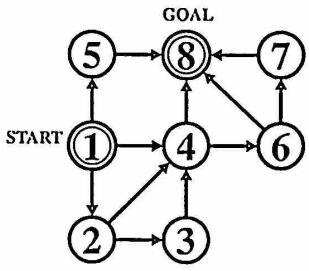

Figure C.1. Sample Input 2