1 Scrat

Task: Perhaps you know Scrat, the sabre-toothed squirrel from Ice Age. Scrat spotted an acorn on the other side of the river. It wants to reach the other side of the river, grab it and then come back to this side. There are ice floes on the rivers, so fortunately Scrat doesn't have to jump from one side to the other, but can jump from ice floe to ice floe. However, there are two types of ice floes: The bigger ones could be used arbitrarily often, but the small ones could be used only one time, then they are broken.

Scrat wants to minimize the length of the maximal jump between the shore and an ice floe, or between ice floes on its way to other side and back again.

Input: There is only one test case specified. The first line contains the number n of ice floes and the distance $D \in \mathbb{N}$ between both shores. Then, n lines follow, which specify the ice floes. An ice floe is specified by a character from $\{s, b\}$, where s denotes a small ice floe and b denotes a big ice floe, and an integer d, that specifies the distance of this ice floe from the shore, where Scrat is at the beginning.

Output: The output is given by a single integer that denotes the length of the longest jump Scrat has to make.

Sample Input 1:

```
3 36
s 4
b 8
b 21
```

Sample Output 1:

```
15
```

Sample Input 2:

```
3 25
s 6
s 7
s 10
```

Sample Output 2:

```
18
```

2 Lazy Teacher

Task: The students in the Lab are unhappy with the workload. Some ask for more difficult tasks while others want easier exercises. To accommodate the students, you decide to let them vote on how the next exercise sheet should look

like. Towards this end you select a group of students that will form a committee of size k and vote on the matter. Every member of this committee has to either vote increase or decrease the workload.

What the students do not know is that you have already prepared the next tasks and because you are lazy, you prefer to not change anything about it. So the best outcome for you is if the vote ends in a tie. You know that every student $i \in [n]$ will vote to increase the workload with probability p_i and vote to decrease workload with probability $(1 - p_i)$.

Choose a subset of the students $S \subseteq N$ with $|S| = k$ such that the probability to get a tie is maximized.

Input: The first line contains the number of candidates n and the size of the committee k. Here, k is guaranteed to be even. After that, each line contains the probability p that the corresponding candidate votes to increase the workload.

Output: For each test case, the output is a single line containing the maximal probability to achieve a tie rounded to 2 decimal places.

Sample Input 1

```
4 2
0.4
0.8
0.3
0.5
```

Sample Output 1:

```
0.62
```

Sample Input 2

```
8 4
0.1
0.3
0.9
0.6
0.5
0.3
0.9
0.5
```

Sample Output 2:

```
0.57
```

3 Longest Increasing Subsequence

Task: In computer science, the longest increasing subsequence problem is to find a subsequence of a given sequence $(a_i)_{i\in[1:N]}$ in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. This subsequence is not necessarily contiguous, or unique. (wikipedia) Compute the length of a longest increasing subsequence.

Input: There is only one test case. In line i the i-th element of the sequence is given. All values are integers. You can assume that the length of (a_i) is upper bounded by 10^9, and $a_i \in [0 : 10^7]$.

Output: Print the length of a longest increasing subsequence.

Sample Input:

```
0
8
4
12
2
10
6
14
1
9
5
13
11
7
15
```

Sample Output:

```
6
```

4 Explosives

Task: You apparently really like to blow up things. How else could you explain that you have prepared several small piles full of fireworks and flour? Now you want to connect these piles with fuzes so that you can watch all of the piles explode one after the other. It would be a shame if two piles exploded at the same time. Therefore your fuzes form a chain without branchings. If you want to lay a fuze between two piles, you need an additional length of 8 cm at each end of the fuze to reach the cores of the piles, where the fireworks are placed. You want to minimize the total length of all fuzes (fuzes are expensive, after all).

Input: The input will consist of several test cases. The first line of each test case contains the number $0 \leq P \leq 14$ of piles. If P is equal to 0, this indicates the end of input. The following P lines contain the (x, y) coordinates of each pile, where each coordinate is an integer between 0 and 150.

Output: For each test case the output consists of a single line containing the total length of all fuzes. All numbers are rounded to 2 decimal places.

Sample Input:

```
6
7 21
57 30
40 103
30 64
113 86
45 118
5
10 26
83 98
141 80
87 29
94 37
3
133 74
50 87
73 112
0
```

Sample Output:

```
305.45
274.40
136.99
```

5 Bipartite or not?

Task: We want to check whether a given undirected graph is bipartite.

Input: The first line of the input contains the number $n \leq 10^6$ of vertices (labelled $0, \ldots, n-1$) and the number $m \leq 10^8$ of edges of the graph. Every following line consists of two vertex labels v and w indicating that there is an edge between v and w.

Output: If the graph is bipartite, print "bipartite". Otherwise, print "not bipartite".

Sample Input:

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

4 3
0 2
3 2
0 3
```

Sample Output:

```
bipartite

not bipartite
```

6 Circular Proofs

Task: Congratulations! You have just finished your P $\neq$ NP proof. But the write-up is a total mess. Up to now, you have thousands of different lemmata, which depend on each other. You have numbered these lemmata from 1 to $n \leq 10000$ in the order in which you have proved them. But for publication you have to rearrange them in such away that every lemma only depends on lemmata that are in front.

Input: The first line of a test case contains the number n of lemmata and the number m of dependencies. The following m lines contain two numbers i and j meaning that lemma j depends on lemma i.

An instance with $n = m = 0$ finishes the input.

Output: For each test case print a line with a valid ordering of the lemmata. If this ordering is ambiguous, output the lexicographically smallest ordering. You can assume that there is a valid ordering, i.e., your proof is not circular.

Sample Input:

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

Sample Output:

```
1 5 4 2 3
```

7 Railway network

Task: We are in the country Euclidia. In Euclidia there are the cities $\{1, \ldots, n\}$ with coordinates in $[1 : 500]^2$. You have to plan the new railway networks. Some cities are already connected, but the goal is that you can travel between arbitrary cities. In order to do this, you establish new railway connections between towns. A new railway connection connects exactly two towns and could be used in both directions. The travel time using a single railway connection is given by the Euclidean distance between the two connected towns.

There are two side conditions for the railway network, that have to be satisfied:

1. To save money, you have to establish no more railway connections than necessary;
2. The travel time using a single railway connection should be minimized.

What is the minimum travel time using a single railway connection that we can obtain?

Input: There will be only a single test case. The first line contains at first the number of towns n, and then the number m of railway connections already established. In the following n lines the coordinates of the towns are given. Afterwards, there are m lines. Each line contains two indices i, j from $\{1, \ldots, n\}$ that means that there is already a railway connection between the towns i and j.

Output: The minimum travel time using a single railway connection rounded up to three decimal places.

Sample Input:

```
3 1
77 288
106 397
489 265
1 3
```

Sample Output:

```
112.792
```

8 Collecting Eggs

Task: After playing basketball, your group of friends is hungry. Fortunately, you are at your grandparent's farm and there are dozens of chickens sweeping around and laying eggs. Because chickens are animals of habit, each chicken has only one specific nest where it lays its eggs. Of course, you know all of these nests because you and your friends exploit your grandparents way too often. So

you plan to split up your group and collect the eggs such that every nest gets visited by at least one of your friends. Afterwards, you meet at the kitchen to make scrambled eggs.

Input: The first line of input contains the number of test cases. The first line of each test case contains the number $N \leq 100$ of locations (that is, the nests, the basketball court, and the kitchen), which are numbered from 0 to $N - 1$. The next line contains the number E of neighborships. The following R lines each contain two distinct numbers $0 \leq v, w < N$, denoting that location v and location w are neighbored. You can travel in 1 minute between neighbored locations. The last line of each test case contains the location indices of the basketball court and the kitchen. It will always be possible to reach every location starting from the kitchen (but not necessarily in one step).

Output: For each test case, the output is a single line containing the case number followed by the minimum time (in minutes) required until you can start cooking.

Sample Input:

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

Sample Output:

```
Case 1: 4
Case 2: 1
```