Problem Set 4

1 Sweet Tooth

You are preparing giveaway presents for your child's birthday party. For this, you have bought ridiculous amounts of various sweets. Now you want to pack a present for each of the children. It is mandatory that each child receives the same amount of sweets to prevent grievance. Thus, you do this: Out of the s sweets, you make c piles for each of the c children, and each of these piles contains $\lfloor s/c \rfloor$ sweets, and then you eat the remaining $s - c \cdot \lfloor s/c \rfloor$ yourself. With this general plan in mind, you set out to make customized piles for all children since each child prefers different sweets.

Input: The first line contains c, the number of children. The second line contains t, the number of different types of sweets. The third line contains t numbers, the ith number says how many sweets of type i you have (this means that s is the sum of the numbers in the third line).

The following c lines each contain t numbers between 1 and c which indicate the preferences of the children for each type of sweet (a lower number indicates that the sweet is higher in the priority list; priorities can be used multiple times).

Output: Compute the minimum p such that it is possible to cluster the sweets into c piles for the children and one pile for yourself such that each child receives $\lfloor s/c \rfloor$ sweets with a priority of $\leq p$.

Sample Input:

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

Sample Output:

```
2
```

2 Express Delivery

Task: You join a company that makes deliveries using smart drones. Each station has a different type of drone with a specific speed and battery capacity. The supply of that type of drone at that station is practically infinite.

Your company does deliveries from one station to another and your task is to find the fastest possible delivery paths. After a drone leaves a station, it can travel until its battery capacity is depleted. At this point, it has to be inside another station to hand over the delivery to another drone. The drones cannot be recharged.

You know the distances between stations, the speeds and capacities as well as a list of possible delivery requests. For each possible request, compute the fastest possible delivery time.

Input: The input has the following form:

- The first line contains two integers: n, the number of stations, and q, the number of deliveries.
- The next n lines contain two integers each: c_i is the capacity in kilometers and s_i is the speed in kilometers per hour of drones at station i.
- The next n lines contain n integers each. The j-th integer on the i-th line gives the distance $d_{i,j}$ from station i to station j in kilometers. The distance -1 indicates that station j cannot be reached directly from station i.
- The next q lines contain two integers each: u_k, the start, and v_k, the end station of delivery request k.

Output: Compute the time y_i in hours required for each request i rounded to two places. The output is a single line $y_1\ y_2\ \dots\ y_q$ given as floating point numbers.

Sample Input 1:

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

Sample Output 1:

```
0.58
```

Sample Input 2:

```
4 3
30 60
10 1000
12 5
```

$20 \;\; 1$
$-1 \;\; 10 \;\; -1 \;\; 31$
$10 \;\; -1 \;\; 10 \;\; -1$
$-1 \;\; -1 \;\; -1 \;\; 10$
$15 \;\; 6 \;\; -1 \;\; -1$
$2 \;\; 4$
$3 \;\; 1$
$3 \;\; 2$

Sample Output 2:

$0.51 \;\; 8.01 \;\; 8.00$

3 Brackets and Question Marks

Task: You are given a sequence that contains three symbols: (,), and ?. The task is to figure out if you can replace the question marks by brackets such that a legal bracket term is generated. A bracket term is legal if at any point in the sequence of (and), the number of (up to that point is greater or equal to the number of) up to that point, and additionally, the total number of (and) is equal.

For this task, you have a computer at your disposal, but it is stone-aged and has very little main memory. It is even overburdened with storing the sequence of brackets and question marks.

Note: Solve this task in C++.

Input: A line containing a sequence of (,) and ?.

Note: Be sure to handle line breaks correctly; each line may have a unix-style (`'\n'`) or a windows-style (`'\r\n'`) linebreak, or (if it is the last line) no linebreak at all.

Output: Output 1 if it is possible to replace all ? such that a legal bracket term is the result, and 0 otherwise.

Sample Input 1:

```
((??))
```

Sample Output 1:

```
1
```

Sample Input 2:

```
((?))
```

Sample Output 2:

```
0
```

4 The Fast and the Furious II

Task: After your last race in Manhattan you appeared in the list of the most wanted people of the FBI. So you decide that it is better to escape from the United States and go to South America. Unfortunately, you also have to leave your loved Lamborghini behind.

The street racing competitions here are similar to the ones in Manhattan: You and your opponent are given some destinations, must visit them all and the one who returns to the start first wins. But there are also some differences: The street network here is different than in Manhattan, usually you have to visit fewer destinations, and your new car is not faster than your opponent's car.

Input: The first line contains three integers n, m and d. n is the number of crossroads, which are numbered from 0 to $n-1$, m is the number of streets, and d the number of destinations (including the starting point). Then, m lines follow, which describe the streets that connect the crossroads: Each line contains three values a_i, b_i and w_i. a_i and b_i are the two crossroads that are connected by this street, and w_i denotes its length. Finally, in a last line the d destinations are described. Here, the first destination is start (and finish) of the race.

You can assume that $1 \leq n \leq 50000$, $d \leq 20$, and all values are integers. Moreover, you can drive on the streets in both directions.

Output: Output the length of a shortest tour beginning and finishing in the start and visiting all destinations.

Sample Input:

```
9 12 3
0 1 2
0 3 5
1 2 4
1 4 2
2 5 2
3 4 5
3 6 1
4 5 4
4 7 4
5 8 2
6 7 3
7 8 5
0 2 8
```

Sample Output:

```
20
```

5 Area of Intersection of Polygons

You are given several simple polygons. Approximate the size of the area where all these polygons intersect.

Input: The first line of the input contains the number of polygons n. Then n lines follow. Each line i starts with the number of corners k_i of polygon i followed by k_i pairs of numbers a_{i,k_i}, b_{i,k_i} that describe the coordinates of each corner. The polygon is constructed by connecting consecutive corners by lines and connecting the last corner to the first corner by a line.

You can assume that the coordinates of the corners are integers, and that it holds true that $0 \leq a_{i,k_i}, b_{i,k_i} \leq 10$.

Output: The size of the area where all n polygons intersect with a rounding error of at most 0.05.

Sample Input:

```
2
4 1 1 4 1 4 4 1 4
3 2 0 6 0 4 2
```

Sample Output:

```
0.5
```

6 Tyrion

During his time as hand of King Joffrey, Tyrion Lannister met Varys, the Master of Whisperers, in the gardens of the palace. ‘Tyrion’, Varys said, ‘I have made a list of all the houses, major and minor, and sorted them according to whispers and rumors, into who is friend and foe of House Lannister. Would you like to know those you are truly on your side?’ Tyrion shrugged, ‘of course’, he says, ‘but I think I have a fairly good idea myself. ‘And he named a few of which he was very certain. ‘Ha!’, Varys said, ‘not bad, but not true either. I reveal this to you: The number of houses that you named and that are true friends of house lannister, this number, it is odd.’ Tyrion was not amused, but then made up his mind, naming a different set of houses. Again, Varys would not tell him which were right, but only that the number of friends in the list was this time even. And so they continued for quite some time, until Varys remarked: ‘But now, my dear friend, you must have gotten it. You should now my true list of true friends.’ Tyrion nodded, and while the listening bystanders were utterly confused, the two men parted ways knowing that they were in absolute agreement about the secret list.

Input: Concise notes of the conversation. It begins with the number of houses in consideration, a number n, in the first line. Then n lines follow. Each is of the following form: First, houses are named, separated by spaces. Then the last word in the line is either ‘even’ or ‘odd’. You may assume that the n lines are enough to tell Tyrion the friendship information for all houses.

Output: Output the number of friends of house Lannister.

Sample Input:

```
4
Baratheon Lannet Stark Targaryan odd
Baratheon Lannet Targaryan odd
Baratheon Stark Targaryan even
Stark Targaryan even
```

Sample Output:

```
1
```

7 Duckburg

Scrooge McDuck (german: Dagobert Duck) runs a new lottery. To participate, players have to pay 25 Dollar. Then each participant chooses an integer between 1 and 1000 and gives it to Scrooge. After all participants have chosen, Scrooge chooses i integers between 1 and 1000. Now all participants get their winnings. For every participant, the closest 'Scrooge number' to the participant's number is determined. The absolute difference between the two numbers is what the participant gets in Dollar.

Write a program that maximizes Scrooge's winnigs / minimizes his losses.

Input: The first line contains $i \leq 100$, the second line contains $n \leq 1000$. Then n lines containing a number between 1 and 1000 each, which is the number that the participant chose. You can assume that all participants choose different numbers.

Output: An integer which is Scrooge's net total if he chooses his numbers optimally. The number is positive if he wins money and negative if he loses money.

Note: We ask for an optimal solution here. If we see a heuristic, we might upload new testcases into the running contest.

Sample Input:

```
10
11
1
2
3
4
5
6
7
8
9
10
11
```

Sample Output:

```
274
```

8 Smooth Operator

You operate a large energy network consisting of sites and links between the sites. Each link has a small chance to fail. Since you mostly stare at old monochrome displays and state-of-the-art Windows NT applications, your thoughts drift off and you wonder about the reliability of the network. From historical data, you know the failure probability of each link in the network. You also know that the entire network breaks down if – as a result of link failures – the network splits into (at least) two disconnected parts. So you come up with the following idea to gauge the reliability: If you look at any subset S of the sites, then the probability that all links connecting sites in S to sites outside of S fail should be less than 10%. Can you figure out which of the networks you operate are reliable?

Input: The first line contains the number n of sites, the second line contains the number m of links. The following m lines contain two integers i and j, $i < j$, and a floating point number $p_{ij} \in (0, 1]$ with three places after the decimal point, thus specifying the failure probability p_{ij} of the link between site i and j. Links with zero error probability are left out, so you may assume that $p_{ij} > 0$. Sites are indexed from $0, \ldots, n-1$.

Output: Output `1` if the network is reliable and output `0` otherwise. The network is reliable if for all subsets S of sites it holds that the combined failure probability $\prod_{i \in S, j \notin S} p_{ij}$ is strictly less than 10%.

Sample input:

```
4
4
0 1 0.1
1 2 0.1
2 3 0.1
0 3 0.1
```

Sample output:

```
1
```

Sample input II:

```
4
4
0 1 0.1
1 2 0.1
2 3 0.3
0 3 0.4
```

Sample output II:

```
0
```