10th Virginia Tech High School Programming Contest

Dec 9, 2023

As a reminder, here are the key rules under which this contest is conducted:

- Teams may not communicate with another human during the contest about the problems.
- Teams may not use more than 1 computer.
- AI assistance (CoPilot, TabNine, ChatGPT) is not permitted.

This problem set contains 11 problems which target a variety of skill levels. You are not expected to solve all of them, particularly if this is your first programming contest!

Enjoy!

Problem A
Best Buy

Steam is digital video game distribution service that is highly popular among PC gamers. You and your group of friends are trying to find a game to play together and everyone is only willing to play the game if it is on Steam. Each of your friends has a list of games they currently own and an amount of money in their steam wallet which they can use to buy a game. Steam also allows players to gift games to other players, and your friends are willing to buy each other a game as long as everyone can play it together. Steam does not allow players to transfer funds from wallet to wallet, so each player gifting a game must be able to afford the game in its entirety. Each game has a name and an associated price and each player's steam account has a list of the games they own and an integer amount of money in their wallet. Each person in your friend group is willing to buy as many copies of a game as it takes to allow everyone to play together. Given a group of friends' steam accounts, find the game that costs your friend group the least net amount of money while still allowing everyone to play the game.

Steam Logo

Input

The first line of input contains an integer, $2 \leq M \leq 10^5$, that identifies the number of games being considered. The second line contains an integer, $2 \leq N \leq 10^3$, the number of friends in your friend group. The next M lines list the name and price of each game, separated by a space. Each name consists of up to 30 uppercase or lowercase English letters. The price of each game is an integer between 1 and 1 000, inclusive.

The following N lines each describe each friend in your group. Each of the lines start with a integer, which is the amount of money in this friend's wallet, followed by a space separated list of names which are the games in the friend's library. Each game is listed at most once. All listed names correspond to an entry in the earlier list of games.

Output

Output the name of the game that should be bought and the total amount of money that needs to be spent to buy it, separated by a space. It is guaranteed that there is at most one game that is the cheapest to purchase overall while ensuring each friend has a copy. If there is no game that all players can play with their existing funds, then output "`no games found`".

Sample Input 1

```
4
3
RocketLeague 15
Dota 8
Apex 20
Valorant 25
0 RocketLeague Dota
15 Apex Valorant
10 RocketLeague
```

Sample Output 1

```
RocketLeague 15
```

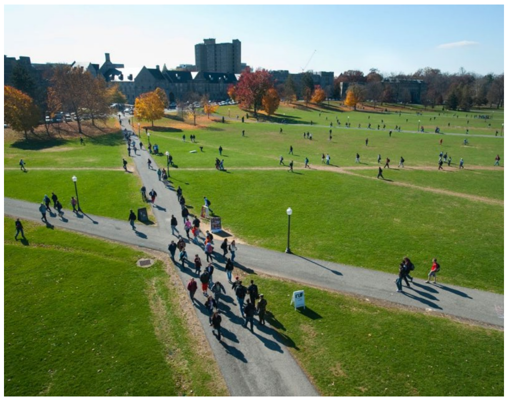
Sample Input 2

```
5
4
A 20
B 10
C 15
D 25
E 30
5 A B D
0 C D
15 E
0 D B
```

Sample Output 2

```
no games found
```

Problem B
Class Scheduling


Drill field at Virginia Tech during class change

John Hokie is a student at Virginia Tech and is working on completing his course request for next semester. This semester he was cursed by having an 8am lecture and a 3 hour 7pm lab. The days of getting on campus before the sun rises and leaving after it sets are surely over next semester! Mr. Hokie knows exactly what classes he needs to take and scoured the timetable to compile all of the section times for each class he needs to take on a particular day.

He desires a schedule that needs him to be on campus for the shortest amount of time. That is, he wants the schedule with the shortest time between the start of his first class and the end of his last class. He's curious what the minimum amount of time he will need to be on campus for is and needs your help to quickly find it out. Keep in mind, though, that he will need at least 15 minutes between classes to have enough time to walk across campus. John is serious about being on time to his classes!

Input

The first line contains an integer n $(1 \leq n \leq 8)$ denoting the number of classes he needs to take. This is followed by n sections, where the first line contains s, k, where s is the name of the class and is composed of up to 8 lowercase letters and numbers, and k is an integer $(1 \leq k \leq 10)$ denoting the number of sections for that class. The next k lines in this section contain two times t_1, t_2, denoting the start and end time of the class section where $(8:00 \leq t_1 < t_2 \leq 20:00)$. Each time is denoted in 24-hour format following the format $h:m$ where leading zeroes in h and m may be excluded. It can also be guaranteed that the duration of all classes is between 45 minutes and 90 minutes and that no two sections for a given class will overlap. Note that due to professors' teaching styles and scheduling constraints, different sections of the same class may be of different durations. Additionally, you can assume there is always at least 1 valid schedule.

Output

Output a single number, which is the minimum number of minutes that John Hokie must stay on campus with the shortest possible schedule that covers all his classes by selecting exactly one section of each class.

Sample Input 1

```
3
cs101 3
13:0 14:15
8:0 8:50
10:0 11:15
cs103 1
9:30 10:30
math334 3
15:0 16:0
12:0 13:0
10:0 11:15
```

Sample Output 1

```
300
```

Problem C
Donut Toppings

Image by anya1 from Pixabay

You own a small donut company. You have n donut boxes of varying sizes and t different toppings that you can put on the donuts in the boxes. You are also given the number of donuts each box holds. You will apply one topping to each donut such that the number of donuts with a certain topping in each box will be within one of another box. In other words, the difference between the number of donuts with any one topping in one box and the number of donuts with the same topping in any other box is at most 1. Other than that, you may use any topping as often or as little as you like.

Input

The input consists of two lines. The first line contains two integers n $(0 < n \leq 1\,000)$ and t $(0 < t \leq 1\,000)$, where n is the number of boxes and t is the number of toppings. The second line contains n integers d_i $(0 < d_i \leq 600)$ separated by spaces, which denote the number of donuts in each box.

Output

If it isn't possible to achieve the desired distribution of toppings, output '`IMPOSSIBLE`'. Otherwise, output n lines. On the i^{th} line output d_i integers that describe which toppings go on the donuts in box i. Donut toppings are numbered from 1 to t, inclusive. If there are multiple solutions, then any of them will be accepted.

Sample Input 1

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

Sample Output 1

```
IMPOSSIBLE
```

Sample Input 2

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

Sample Output 2

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

This page is intentionally left blank.

Problem D
Late Halloween

Source: Pixabay

It's currently Halloween and there are three children trick-or-treating at your house. The problem is that it's getting late, and you want to hand out the rest of your goodie bags. You have several bags, but you didn't have time to count out candies, so each bag has a varying number of candies. Return whether it is possible to partition the bags so that each child has the same number of candies and no bags are leftover.

Input

The input begins with an integer n $(1 < n < 1000)$ which denotes the number of goodie bags. The next line contains n space separated integers c_i $(1 < c_i < 100)$ that describe the number of candies in each goodie bag. The sum of all c_i is guaranteed to be less than $3\,000$.

Output

Output `True` if you can fairly distribute the goodie bags, otherwise, output `False`.

Sample Input 1

```
10
9 7 3 5 1 4 2 1 4 3
```

Sample Output 1

```
True
```

Sample Input 2

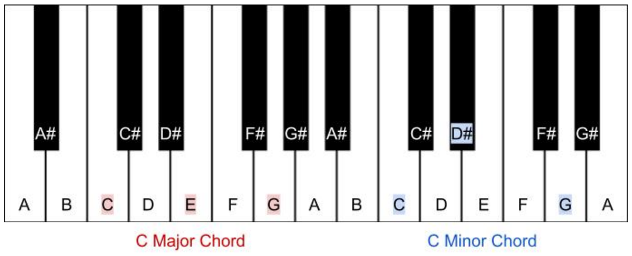
```
5
2 2 2 1 2
```

Sample Output 2

```
False
```

This page is intentionally left blank.

Problem E
Major or Minor



C Major and Minor chord shapes

You are a student at a university that combines music and technology. As a part of the curriculum, you are assigned to write a program to recognize basic chord shapes on the piano.

Neighboring keys on a piano (including both black and white keys) correspond to notes that are a semitone apart. The leftmost key on a standard piano keyboard corresponds to a note called A. A semitone up from A is note $A\#$. Note B is a (whole) tone above A, which is two semitones up from A. If we continue to add semitones, we obtain notes C, $C\#$, D, $D\#$, E, F, $F\#$, G, $G\#$ before the pattern repeats. To distinguish notes with the same name that are separated by a multiple of 12 semitones (also called an octave) a subscript is used: for instance, A_1 is 12 semitones up from A_0.

To represent the piano digitally, we represent each key on the piano by its distance from the lowest note on the piano. Thus the leftmost piano key, A_0, corresponds to 0. B_0 corresponds to 2. A_1, the piano key one octave above A_0, corresponds to 12, and so on.

A musical interval measures the number of semitones between any two notes. For instance, a so-called major third interval spans 4 semitones, a minor third spans 3 semitones, and a perfect fifth encompasses 7 semitones.

A (triad) chord contains 3 notes: a root note, a note that's up a major or minor third (up 4 or 3 semitones, respectively), and a third note that is a perfect fifth (seven semi-tones) from the root note.

Write a program that, given a set of three musical notes, outputs whether these notes are part of a major or minor chord. To determine whether notes are part of a chord, ignore the octave to which they belong: for instance, (C_0, E_4, G_2) would still be considered a C major chord since it contains C, G, and E.

When identifying chords, your program should output the root note and whether the chord is major or minor.

Input

The input is a single line with three integers a, b, c $(0 \leq a, b, c \leq 87)$ corresponding to three distinct notes on a piano. The notes can appear in any order.

Output

Print the root note and whether the chord is major, minor. If the three notes do not correspond to a chord, output `neither`.

Sample Input 1	Sample Output 1
`0 4 7`	`A major`

Sample Input 2

```
1 5 8
```

Sample Output 2

```
A# major
```

Sample Input 3

```
0 3 7
```

Sample Output 3

```
A minor
```

Sample Input 4

```
1 4 8
```

Sample Output 4

```
A# minor
```

Sample Input 5

```
11 2 6
```

Sample Output 5

```
G# minor
```

Sample Input 6

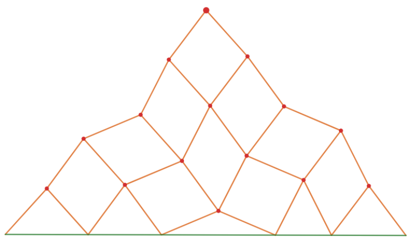
```
87 86 83
```

Sample Output 6

```
neither
```

Problem F
Matchstick Men

A group of $n^2 - n$ identical matchstick men are building a matchstick structure where $2n - 2$ matchsticks are lined up on a horizontal line with n marked points, with one matchstick propped up at either end and 2 matchsticks each at the interior points. The tops of each pair of matchsticks touch (and melt into a single point). These matchsticks then form $n - 1$ triangles, and another group of matchstick men will build on top of these triangles' top vertices until a single pair of matchsticks touches to form the top of this structure. Given the placements of the men at the bottom, how high will their structure be?


Matchstick Men

Input

The first line of input contains an integer n ($2 \le n \le 50$), followed by a real number t ($1 \le t \le 10$), which is the length of each matchstick. The second line of input contains $n - 1$ real numbers a_i where a_i is the distance between the i^{th} and $(i+1)^{\text{th}}$ matchstick man. It is guaranteed that $2t > a_i$ for all values a_i. All real numbers will have no more than 3 digits after the decimal point.

Output

Output the height of the matchstick structure. Your answer will be considered correct if it is within an absolute error of 10^{-6} of the real answer.

Sample Input 1

```
2 2
2
```

Sample Output 1

```
1.73205080756887720
```

Sample Input 2

```
6 5
5 4 7 3 5
```

Sample Output 2

```
21.58323995415637600
```

Sample Input 3

```
6 5
5 4 7 2.985 5
```

Sample Output 3

```
21.58559211845467000
```

Problem G
The Seventh Dragon Ball

Source: Pixabay

In an alternate Dragon Ball universe, Earth has been destroyed. Luckily, you are among a group of survivors who made it out to a new planet with six dragon balls and a mysterious machine sealing the seventh dragon ball. If you unlock enough levels of the machine, you will have all seven dragon balls needed to summon Shenron and wish for Earth back!

From your observation and reading the machine manual, you deducted that:

- The machine operates on the concept of "Ki" (energy). Each level requires a specific amount of Ki to unlock, starting from 1 Ki for the first level, and each subsequent level requires 1 unit of Ki more than the previous one. Levels must be unlocked one by one.
- To unlock a level, the sum of Ki transferred by the survivors must exactly match the required Ki for that level.
- The machine immediately absorbs all the Ki from each person who attempts to transfer.
- Once a person transfers their Ki to unlock a level, they cannot transfer again for that particular level due to slow Ki recovery. Luckily, each survivor regains all their Ki for subsequent levels.
- You discovered a magic code to unlock a level without matching the exact Ki requirement. However, you may use this code only once, that is, for exactly one level.

Output the maximum number of levels you can unlock, including possibly using your magic code.

Input

The input consists of:

- The first line with one integer k ($1 \leq k \leq 1\,000\,000$) represents number of people on the new planet.
- The following k lines each contain an integer k_i ($1 \leq k_i \leq 2\,147\,483\,647$) representing number of Ki the i^{th} person possesses.

Output

Output the maximum number of levels you can unlock.

Sample Output Explanation

In Sample Input 1, it is possible to unlock levels 1 through 11 by combining Ki and use the magic code on 12. Thus, 12 levels can be unlocked. In Sample Input 2, it is possible to unlock levels 1 through 3 by combining Ki, use the magic code on 4, and unlock levels 5 through 8. Thus, 8 levels can be unlocked in total.

Sample Input 1

```
4
1
2
3
5
```

Sample Output 1

```
12
```

Sample Input 2

```
3
2
1
5
```

Sample Output 2

```
8
```

Problem H
Step Goals

Hiking trail on a mountain. Source: Pixabay

Alice is an avid walker and always makes sure to reach her daily steps goal. She is always looking for new places to go on walks, and prefers nearby mountains which all have connected trails. Depending on how much activity she has already gotten on a particular day, Alice will have a different number of steps needed each day to meet her daily goal.

The mountains near Alice are no ordinary mountains. They appear to have no start and end, and are oddly shaped. Furthermore, these mountains all have connecting trails, which means that the end point of a trail is the start point of the next trail. Fortunately, Alice has a teleportation device that can take her to the start point of any trail and back from the trail's end point! However, Alice can use the device she has only twice a day, once to teleport to a trail's start point, and once to get back from the end point of the last connected trail she takes to the bottom of the mountain after she is done.

Alice wants to make sure she starts at a trail such that she can walk on the least number of trails possible while still meeting her step goals. Alice must keep walking in the same direction; she cannot turn around and repeat a trail to reach her step goal. Thus, she will continue walking until she meets her step goal, at which point she will teleport back to the bottom of the mountain.

Given the number of steps she wants to take that day and the connected trails with their distances, find the minimum number of consecutive trails she needs to walk to meet her steps goal.

Input

The first line of input will contain 2 integers. The first integer s ($1 \leq s \leq 10^9$) is the number of steps Alice will walk that day, and the second integer n ($1 \leq n \leq 10^5$) is the number of trails that Alice can walk on. The second line will list n integers t ($0 \leq t \leq 1\,000$) which provide the lengths of the connected trails (in order) in miles.

For Alice, 1 step $= 1$ foot. There are $5\,280$ feet in a mile.

Output

Display the minimum number of trails Alice will have to take to meet her steps goal for the day. If it is impossible for her to meet her step goals, then output `IMPOSSIBLE`.

Sample Input 1	Sample Output 1
22000 6 2 3 1 2 4 3	2

Sample Input 2

```
45000 3
3 1 1
```

Sample Output 2

```
IMPOSSIBLE
```

Problem I
3019

The year is 3019, and the world is experiencing a pandemic named COVID-19 (which, for some reason, sounds hauntingly familiar). Tragically, it is transmitted through tropospheric air and very potent, so the various communities of the world have transitioned to living underground in n underground bases, conveniently named using the integers from 1 to n. In order to share resources between each other, humanity has developed m one-way portals, which each have a start point in one base and an endpoint in a different base.

Fortunately, it was recently made public that a brilliant scientist has created a cure for COVID-19! Although the cure has not started being industrially manufactured as of yet, it is a product of such importance that the plan for its distribution is already known: autonomous drones will begin at each production facility and will travel from base to base, delivering the cure. In fact, the first facility for its mass-production is already underway as well. More will inevitably come later, but it would be preferable for the first one to be located at a base which allows a single drone that starts there to reach a maximal number of bases **in a single trip** (by transporting the cures across bases through some series of portals, possibly passing through some bases or portals more than once). Given the layout of bases and portals, find a set of bases that can all be reached from a single base by going through some route of portals such that this set of bases has maximum size.

Input

The input consists of a single test case. The first line contains two integers n and m ($1 \leq n, m \leq 100\,000$) where n denotes the number of bases and m denotes the number of one-way portals. Each of the following m lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq n$), where $i \in \{1, 2, \ldots, m\}$, which denote the bases at the start point and endpoint of portal i, respectively.

It is guaranteed no ordered pair of two bases is directly connected by more than one portal and that no portal has its start point and endpoint within the same base.

Output

Output three lines. The first line should specify the maximum number of bases that meet the conditions specified in the problem. The second line should print all of those bases in any order, including the start base. Although the drone is allowed to enter the same base multiple times during its trip, each should be printed only a single time. Leave spaces between all integer numbers on the same line. The third line should specify the base at which the production facility should be located.

If there are multiple possible solutions, you may print any of them.

Sample Input 1

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

Sample Output 1

```
2
2 3
2
```

Sample Input 2

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

Sample Output 2

```
3
1 4 5
1
```

Sample Input 3

```
6 6
1 2
2 3
3 4
4 5
5 6
6 1
```

Sample Output 3

```
6
2 3 4 6 1 5
4
```

Sample Input 4

```
19 23
1 10
10 9
9 8
8 7
7 3
3 1
10 2
8 12
11 6
11 12
6 11
17 4
4 5
17 16
13 14
14 15
13 19
2 12
16 15
19 6
15 13
13 17
17 18
```

Sample Output 4

```
9
6 11 12 13 14 15 16 17 19
13
```

This page is intentionally left blank.

Problem J
Treasure Trouble

When Theta was younger, one of her favorite games was "Treasure Trouble," a boardgame for children 5 and up, which is made by Noris-Spiele. In this game, players are pirates whose goal it is to collect treasures. At the beginning of each round, an hourglass is turned over and the youngest player shouts: "For the Cap'n," after which the players start filling their treasure chests with loot of various shapes, colors, and sizes before the hourglass runs out. After the time has run out, each chest is carefully checked to make sure its lid lies flush on top — any items too large to fit must be removed. Each player receives 1 point for each item they were able to place into their chest.

There is a twist, however: before the points are counted, a card (chosen from a deck of cards) is flipped over. This card shows treasures of various shapes, colors, and sizes. All treasures that are listed on the card must be given to Jewel Jack (and thus do not count for the player who put them in their chest!)

Given a set of cards, find the optimal strategy for one player to fill their chest such that the expected number of points is maximized!

Input

The input consists of a single test case. The first line of this test case contains three integers S ($1 \leq S \leq 1000$), T ($1 \leq T \leq 40$), and C ($0 \leq C \leq 25$). S is the size of the treasure chest, T is the number of available treasures, and C is the number of cards that may be drawn. Treasures are numbered from $1 \ldots T$. The next line contains T integers denoting the sizes of the available treasures s_i ($0 < s_i \leq 1000$).

This is followed by C lines, one for each card. Each line starts with an integer k that denotes the number of integers to follow. Each integer specifies the number of a treasure that is displayed on that card (and which does not count for the player if this card is drawn).

Output

Output a selection of treasures such that the expected number of points is maximized. Refer to each treasure using its number ($1 \ldots T$). You may output the treasures in any order. If there is more than one selection of treasures that maximizes the expected gain, you may output any of them.

Sample Input 1 Explanation

In Sample Input 1, there are 4 items from which to choose to place into a chest of size 50. Treasure #1 (of size 10) is listed on cards 2 and 3, treasure #2 (of size 20) is listed on cards 1, 2, and 4, treasure #3 (of size 30) is listed on card 1, and treasure #4 (of size 40) is listed on cards 3 and 4. Based on their sizes, we can place treasures 1, 2, 3, 4, $1+2$, $1+3$, $1+4$, or $2+3$ in the chest (where $+$ denotes ‘and’). The table below shows how many points we can expect for each choice.

We choose	1	2	3	4	1+2	1+3	1+4	2+3	
If card #1 is drawn, we get	1	0	0	1	1	1	2	0	points
If card #2 is drawn, we get	0	0	1	1	0	1	1	1	points
If card #3 is drawn, we get	0	1	1	0	1	1	0	2	points
If card #4 is drawn, we get	1	0	1	0	1	2	1	1	points
Expected number of points	.5	.25	.75	.5	.75	1.25	1	1	

Thus, the choice that maximizes the expected number of points is $1+3$.

Sample Input 1

```
50 4 4
10 20 30 40
2 2 3
2 1 2
2 1 4
2 2 4
```

Sample Output 1

```
1 3
```

Sample Input 2

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

Sample Output 2

```
3 4 5 7 8
```

Problem K
Urban Explorer

Explorer getting chased by guard

You are an urban explorer who has run into a security guard on one of your late night excursions, where you parachuted into a maze of rooms. Luckily you have a map of all of the rooms, how they are connected, and the location of the exit. Additionally, you have one other thing that has kept you safe on your adventures, you have super hearing good enough to pinpoint the exact room that the security guard is in as he chases you. Unfortunately, the security guard has CCTV footage and knows the exact layout of all rooms and where you are as well. You luckily saw the guard first so you are one move ahead of the guard.

Assuming both you and the security guard move optimally and at the same speed, print out whether the guard will catch you, whether you will escape, or if there will be a stalemate. A stalemate occurs when the guard cannot catch you, but you also cannot reach the exit room without getting caught.

Thus, you and the security guard move in turns and during that turn whoever's turn it is must move to one of the rooms adjacent to their current room. For example, if the security guard is in room 1 that is connected only to room 2 and it is their turn, they must move to room 2. Additionally, the security guard is not allowed to occupy the room with the exit.

This game may end in one of two ways:

1. If the security guard ever occupies the same room as the urban explorer, the security guard wins and catches the explorer.
2. If the urban explorer ever reaches the exit room, the urban explorer wins.

If under optimal movement by both you and guard neither situation occurs, you are stuck in a stalemate.

Input

On the first line of input, you are given an integer m ($3 \leq m \leq 50$) representing the number of rooms. This line is followed by m lines with each line representing a room, where the rooms are numbered from 0 to $m-1$. On the i^{th} line there is an integer n ($0 \leq n \leq m$) denoting how many rooms are connected to room i. This is followed by n integers i_k ($0 \leq i_k < m$ and $k \in \{1 \ldots n\}$) on the same line representing the numbers of the rooms to which room i is connected. You may assume that there are no one-way passages between rooms, that is, if there is a passage from room i to room j then there is also a passage from room j to room i. The exit is always room 0. The urban explorer always starts at room 1, and the guard starts at room 2.

Output

Print out "`Caught`" if the guard catches you, "`Escaped`" if you can reach the exit before the guard, and "`Stuck`" if it is a stalemate.

Sample Input 1

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

Sample Output 1

```
Escaped
```

Sample Input 2

```
6
2 2 5
1 3
3 0 4 5
3 1 4 5
2 2 3
3 0 2 3
```

Sample Output 2

```
Stuck
```

Sample Input 3

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

Sample Output 3

```
Caught
```