TeamsCode Spring 2018 MIHS Programming Contest

Problem Set

Rules:

1. Each question is worth 50 points. With each incorrect submission, the value of that question decreases by 5 points. You do not need to solve the problems in order.

2. The internet may not be accessed during the contest coding phase. Only one computer per team can be out during this time, meaning phones also need to be put away.

3. The teams with the highest score at the end of the contest will receive awards. In the case of a tie, the team that made their final successful submission first will be the winner.

Problems:

1. Frog
2. Bookcase
3. Line Graph
4. Chopping Trees
5. Bunny Island
6. Digit Search
7. Lost Phone
8. Intersecting Lines
9. Roman Numerals
10. Reverse Engineering
11. Swap
12. Fast Travel
13. 24 Game
14. Geometric Sequence
15. Factorization

1. Frog
Input File: frog.txt

In *Frog*, you must print out the frog shown below. There is no input in this problem, but your frog must match **exactly** with the one shown, down to the last dots and spacing.

Input:

None.

Output:

You will output the frog exactly as shown below.

Example Output:

```
  @..@
 (----)
/ >__< \
^^ ~~ ^^
```

2. Bookcase
Input File: bookcase.txt

In *Bookcase*, a student has been assigned to organize a row of books. The books are to be arranged alphabetically, but they are currently all scrambled. You must check the book titles and organize them alphabetically. A “space” counts as the earliest character, so “`This Test`” should be arranged before “`ThisTest`”.

Input:

The first line contains an integer N. There are N following books, each book being denoted by its title. No two books will have the same title. None of the book titles will contain an `int` (like `1984`).

Output:

You must output the new arrangement of the books in alphabetical order. Each book should be on a new line.

Example Input:

```
7
The Hunger Games
Ready Player One
Animal Farm
A Short History of Nearly Everything
Dracula
A Thousand Splendid Suns
Dune
```

Example Output:

```
A Short History of Nearly Everything
A Thousand Splendid Suns
Animal Farm
Dracula
Dune
Ready Player One
The Hunger Games
```

3. Line Graph
Input File: line.txt

You are currently learning how to graph lines in Algebra class using the formula y = mx + b. Given the m, x, and b values, your job is to find the y value.

Input:

The first line contains an integer N. The following N lines contain three space-separated integers representing m, x, and b, respectively.

Output:

Output the y value on a separate line for each test case. Your answer will be an integer.

Example Input:

```
3
2 3 4
-10 0 8
5 -5 -6
```

Example Output:

```
10
8
-31
```

4. Chopping Trees
Input File: chopping.txt

You, a lumberjack, are cutting down a small forest. However, the sun is going down, and you must return home once dark falls. Using the rules given below concerning the chopping time, determine the amount of time it takes for you to cut down your small forest (this amount of time will always be an integer; no rounding necessary).

There are three main components of tree-chopping time in this problem:

1. The amount of time it takes to make one swing (between 1-20 seconds)
2. The number of swings to cut down a tree (between 1-20 swings)
3. The clean up time for each chopped tree (between 1-20 seconds)

Given the number of trees in your small forest (between 1 and 250) and values for the three components above, find the amount of time it takes to completely chop down and clean up the small forest.

Input:

The first line contains an integer N. Each of the following N lines will each contain four space-separated integers, the first being the amount of time it takes to make one swing, the second being the number of swings to cut down a tree, the third being the clean up time for each chopped tree, and the fourth being the number of trees in the forest.

Output:

You will output the amount of time, in seconds, it takes for you to cut down your small forest for each test case. Your result should be a non-rounded integer.

Example Input:

```
3
8 3 2 15
13 14 5 130
3 7 9 76
```

Example Output:

```
390
24310
2280
```

5. Bunny Island
Input File: bunny.txt

Ōkunoshima, an island in Japan, is home to thousands of bunnies. Due to a lack of predators, the bunny population has grown rapidly, and can be modeled by the Fibonacci Equation. This equation is shown below:

$$F_n = F_{n-1} + F_{n-2}$$

Your job is to find the new number of bunnies based on two starting numbers (F_{n-1} and F_{n-2}) and a period of time. Given that the bunny population is modeled so that *n* increases by **one every month**, use the given data to find the new bunny population in different scenarios.

Input:

The first line contains an integer N. The following N lines contain three space-separated integers representing F_{n-2} ($1 < F_{n-2} < 100$), F_{n-1} ($1 < F_{n-1} < 100$), and a time period in months ($1 < \text{months} < 25$).

Output:

You will output the new bunny population for each of the different scenarios (each answer gets a new line).

Example Input:

```
4
5 8 20
2 3 1
13 21 5
1 2 16
```

Example Output:

```
121393
5
233
4181
```

6. Digit Search
Input File: digit.txt

You are given a string consisting of several random letters and digits. However, you are only interested in the digits and want to find the highest digit (up to `9`) that appears in the string that is preceded by all smaller digits. These digits must appear in order, from smallest (`0`) to greatest, but do not have to adjacent to each other. For example, in the string `j012ty43a4p5q7s68`, the correct highest digit would be 6 because it is preceded by 0, 1, 2, 3, 4, and 5 in order. 7 does not count because it isn't preceded by a 6, and therefore 8 also does not count. For each input string, find the highest digit.

Input:

The first line contains an integer N. The following N lines each contain a single string.

Output:

Output the highest digit according to the rules listed above. If there isn't a highest digit, print `None`.

Example Input:

```
3
001om525hi43Woe7ih5f
Qwa01qe2a34Gl56jj87qw89ef
ABC98765432123456789XYZ
```

Example Output:

```
3
9
None
```

7. Lost Phone
Input File: lostphone.txt

You were going about your daily routine when you suddenly realize you have lost your phone. Using an app on your computer you know that your phone is a distance `d` away, but unfortunately, the app is terrible and does not tell you the actual location of the phone. You know of multiple places you could have lost your phone, and your job is to determine which of these places match the distance `d` which is mentioned by your app. All coordinates given during this problem are based on the Cartesian coordinate system.

Input:

First, you will be given the distance `d`, representing the distance between you and your phone. On the next line, you will be given your position in terms of `(x, y)` coordinates. After that, you will be given a new line containing an integer N, representing the number of following places you could have lost your phone. These next N lines will each contain the name of the location along with its position in `(x, y)` coordinates. `x` and `y` are both single-digit integers (`1` through `9`).

Output:

You will output the name of the location at which you lost your phone, using the same capitalization.

Example Input:

```
5
(5, 6)
4
Library (3, 9)
School (1, 2)
Home (9, 3)
Football Stadium (9, 2)
```

Example Output:

```
Home
```

8. Intersecting Lines
Input File: intersect.txt

You've now advanced to Algebra II and would like to automate simple Algebra I tasks like finding the point of intersection between two points. Your goal now is to create a program that takes in two equations in the form y = mx + b and prints out the x and y values of the intersection.

Input:

The first line contains an integer N. The following N lines contain two space-separated numbers representing m_1, b_1, m_2, and b_2, respectively.

Output:

Output x and y separated by a space for each test case. Round x and y to the nearest integer. If the lines are parallel and do not intersect, print `None`. If there are an infinite amount of points of intersection (both lines are the same), print `Same Line`.

Example Input:

```
4
2 3 -5 -11
0 3 1 6
2 10 2 -7
17 -1 13 -13
```

Example Output:

```
-2 -1
-3 3
None
-3 -52
```

9. Roman Numerals
Input File: roman.txt

Your task is to create a program that can convert numbers into Roman numerals. For example, if the input given is 7, your program will produce VII, and if the input is 99, your program will produce XCIX. The table below gives the pairings for different values:

I = 1	V = 5	X = 10	L = 50	C = 100	D = 500	M = 1000

Therefore, you can see that to represent 124 in Roman numerals, you would need one C and two X's to form 120. Then, since the remaining 4 is one less than 5, it is represented as IV, signifying V - I = 5 - 1 = 4. Added together, 124 is represented as CXXIV.

Other special cases include 9 = IX, 40 = XL, 90 = XC, 400 = CD, and so on. In addition, the number 4000 is represented in Roman numerals as MMMM.

Input:

The first line contains an integer N. The following N lines each contain an integer between 1 and 4999.

Output:

Output the Roman numeral for each given input. Note: Use capital i rather than lowercase L to represent 1 in Roman numerals.

Example Input:

```
4
46
19
392
88
```

Example Output:

```
XLVI
XIX
CCCXCII
LXXXVIII
```

10. Reverse Engineering
Input File: reverse.txt

You are a construction worker who is collecting materials. You have been given an image of the latest shipment; however, the categories of materials have been mixed up. Your job is to determine the shape, size, and substance type of the various materials. There are a few key pieces to this problem:

- There are three different shapes that the materials can have: **linear**, **triangular**, and **rectangular**.

- Area is calculated using `width * height` for linear and rectangular figures and `0.5 * width * height` for triangular figures, **not** by counting the enclosed spaces.

- The linear shapes will always have a height of `1`.

- Triangular shapes have heights represented by the number of lines and widths represented by the number of characters on the last line.

- The top and bottom lines of the triangle will contain an odd number of characters (excluding spaces), and will always resemble an isosceles triangle similar to the structure below:

```
   X
  X X
 X   X
XXXXXXX
```

- The rectangular shapes can have equal or different heights and widths.

- The minimum width is `2`, the maximum width `10`, the minimum height is `1`, the maximum height is `10`.

- There are three different types of materials, symbolized by their corresponding symbols:
    - **Wood:** `W`
    - **Metal:** `M`
    - **Clay:** `C`

Input:

The first line contains an integer N. There are N following “materials,” each material separated by a new line. The size of the material is symbolized by the width/height of the characters, and the type of material is symbolized by the characters (`W`, `M`, `C`) outlining the shape.

Output:

You will output N lines of descriptions, each line containing the shape, the area, and the substance of the material, in that order. Each piece of data (shape, area, and substance) will be separated by a semicolon and a space. The final area will always be an integer.

Example Input:

```
4
WWWWW
W   W
W   W
WWWWW

   M
  M M
 M   M
MMMMMMM

CCCCCCCCCC

MM
MM
```

Example Output:

```
Rectangular; 20; Wood
Triangular; 14; Metal
Linear; 10; Clay
Rectangular; 4; Metal
```

11. Swap
Input File: swap.txt

The new hit board game *Swap!* has just hit the market. In the game, you are given a rectangular board filled with `X` pieces and `O` pieces. This board, which has unique placement of the `X` and `O` pieces, corresponds to an image of a board with a **different** arrangement of `X` and `O` pieces.

Your job is to make your board match the board in the image in the **fewest possible steps**. However, you can only move the pieces by **swapping** a piece with an adjacent piece (horizontal or vertical adjacent swaps, no diagonal swap). Given your board and the image board, find the fewest number of steps it takes to make the boards match (you can guarantee there is a solution; i.e., each board has the same ratio of `X` to `O` pieces).

Input:

The first line contains two space-separated integers representing the width, w, and height, h, respectively, of the two boards (`1 < width, height < 20`). The next h lines contain your board, followed by another set of h lines containing the image board.

Output:

You will output an integer representing the fewest possible steps to make your board match the image board.

Example Input:

```
5 5
XOOXO
XOXXX
XXOOO
OOOOO
XOXOX
XOXOO
XOXOX
OOOOO
OOOXX
XXOXX
```

Example Output:

```
11
```

12. Fast Travel
Input File: fasttravel.txt

You are a hiker walking through the wilderness. You are running out of food and supplies and thus must reach the cabin as quickly as you can. There are multiple different paths; however, the various paths are very different (**dirt path**, **river**, **mountainous trail**, and **swamp**). You travel at different speeds in each of these path types; the corresponding speeds are listed below:

- **Dirt Path:** 5 meters/second
- **River:** 1 m/s
- **Mountainous Trail:** 2 m/s
- **Swamp:** 2.5 m/s

Your job is to find the quickest path (the smallest amount of time it takes to travel to the cabin).

Input:

The first line contains an integer N. Each of the following N test cases contain the width, w, and height, h, of the maze on the first line and the maze itself on the following h lines.

You will be given a rectangular maze (`1 < width, height < 20`) in ASCII characters. The maze will be made up of the following symbols:

- `#` – A border to the maze. Impenetrable. The maze border will be surrounded by this material, not including the starting and ending points.
- `P` – The start to the maze (your starting position).
- `C` – The position of the cabin (the end destination).
- `D` – A dirt path.
- `R` – A river.
- `M` – A mountainous trail.
- `S` – A swamp.

Each of the different path types (`D`, `R`, `M`, and `S`) is 10 meters long per character (so `DDD` is 30 meters of dirt path).

Output:

Output the time it takes to travel to each cabin in seconds, calculated by how many `D`, `R`, `M`, and `S` positions you must cross.

Example Input:

```
2
10 7
##C#######
#RRD###S##
#DRR#SSSS#
#SRS#SDRM#
#RMMMSSDD#
#DRMMDSDD#
#######P##
7 5
###C###
#SRD#S#
#S##RM#
#MRDMD#
####P##
```

Example Output:

```
48
42
```

13. 24 Game
Input File: 24game.txt

24 Game is a classic math game where, given four positive integers, you must use addition, subtraction, multiplication, division, and/or parentheses to produce 24 from all four numbers. For example, given 2, 3, 4, and 5, it is possible to produce 24 by doing `2 * (3 + 4 + 5)`.Your job is to write a program that determines whether it is possible to create 24 from four input numbers.

Input:

The first line contains an integer N. The following N lines each contain four space-separated, positive integers.

Output:

If it is possible to form 24 from the numbers using addition, subtraction, multiplication, division, and parentheses, print `Possible`. If not, print `Not Possible`.

Example Input:

```
4
2 5 7 3
7 13 19 2
101 50 7 33
24 10 53 5
```

Example Output:

```
Possible
Not Possible
Not Possible
Possible
```

14. Geometric Sequence
Input File: geometric.txt

A geometric sequence is a sequence where each progressive term is found by multiplying the previous term by a constant value. A geometric sequence could be 1, 2, 4, 8, ..., where the common ratio between is `2`, or it could be 120, 60, 30, 15, ..., where the common ratio is `0.5`. For this problem, you can assume that the common ratio is a positive integer. Your task is to find the length of the longest geometric sequence from a given sorted array of integers. The terms of the sequence do not need to be adjacent to each other.

Input:

The first line contains an integer N. The following N lines can each contain anywhere between 1 and 20 integers, each separated by a space. These integers are guaranteed to be sorted from least to greatest.

Output:

For each test case, output an integer representing the length of the longest geometric sequence that can be formed from the given integers.

Example Input:

```
3
1 2 3 4 5 6 7 8 9 10
2 3 4 5 8 9 10 16 20 27 40 81 120 243
3 5 7 11 23 31 101
```

Example Output:

```
4
5
1
```

15. Factorization
Input File: factor.txt

Factoring is always a hassle. As a result, you have decided to implement your own program that can find all the real zeros of a linear or quadratic function. This program takes input in the form $c_1x^{p_1} + c_2x^{p_2} + ...$, where c and p are integers and p is between 0 and 2. If any real zeros exist, your job is to print them out.

Input:

The first line contains an integer N. The following N lines each contain an expression in the form shown above.

Output:

Output the zeros for each test case in sorted order. For integer values, do not include trailing zeros, and for decimal values, round down to three decimal places if any are longer than that. If none exist, print `None`.

Example Input:

```
3
1x^1 + 3x^0 - 2x^1
4x^2 + 9x^1 + 5x^0
19x^2 - 6x^0 - 11x^2 - 8x^2
```

Example Output:

```
3
-1.25 -1
None
```