COMPUTER SCIENCES

George Fox University
H.S. Programming Contest
Division - I
2018

General Notes

1. Do the problems in any order you like. They do not have to be done in order
(hint: the easiest problem may not be the first problem)

2. Scoring: The team who solves the most problems in the least amount of time with the least submissions wins. Each wrong submission will receive a 20 min time penalty that will only be added to the time score once the problem has been successfully solved. Time is calculated for each problem as the total time from the start of the contest to the time it was solved.

3. There is no extraneous input. All input is exactly as specified in the problem. Integer inputs will not have leading zeros.

4. Your program should not print extraneous output. Do not welcome the user. Do not prompt for input. Follow the form exactly as given in the problem.
(hint: spaces? No spaces? What does spec say!)

5. All solutions must be a single source code file.

This page intentionally left blank

A. Go Fish PINK

My daughters love to play the card game "Go Fish." The object of the game is to guess what cards your opponent has so you can match cards in your hand. You start with seven cards. If you guess a card correctly, they hand you that card and you go again. If they do not have the card, they say, "Go fish," and you draw a card from the pile. If you draw the card you guessed, you get to ask again. Otherwise, it's your opponent's turn. Write a program for a SIMPLIFIED game of "Go Fish"!

In this simplified version, there will be only 5 turns (10 guesses total). For each guess, output either "HERE'S MY CARD" or "GO FISH". A player will not go again if they guess the card. Assume you will not take all the cards from your opponent. You will not "draw a card" if you guess incorrectly, so your hand will only get smaller with each correct guess.

Input: There will be 12 lines of data. The first 2 lines contain seven cards (2-10, J, Q, K, A) for the 2 hands. The next 10 lines indicate 5 turns of each player guessing a card, the first player guessing from the 2nd players hand, etc.

Output: For each guess, show either "HERE'S MY CARD" or "GO FISH".

Example Input

```
2 4 6 8 J Q K
3 4 5 6 9 10 A
2
6
8
9
J
A
4
5
Q
3
```

Output to screen:

```
GO FISH
HERE'S MY CARD
GO FISH
GO FISH
GO FISH
GO FISH
HERE'S MY CARD
GO FISH
GO FISH
GO FISH
```

This page intentionally left blank

B. Car Counter YELLOW

You might have seen a strip of cable across the road that DOT uses to count how many vehicles pass by that road each day. It counts each change in pressure as an axle. Small (car), medium (pickups and vans) and large (trucks) vehicles have different axle patterns so that the DOT can distinguish what kind of vehicles pass by in addition to the number of each. Write a program that will emulate this car counter.

For this program, there will be a continuous string of characters (split up in 10 lines of 50 characters each) in which "x" will represent space between "bumps" and the "o" will represent a "bump" of an axle. Small vehicles will have the pattern "oo" surrounded by any number of x's. Medium vehicles will have the pattern "oxo" surrounded by x's. Large vehicles will have the pattern "oxoxxooo".

For example, the following represents 2 small vehicles, followed by 2 medium vehicles, and lastly one large vehicle:

```
xooxxxxooxxxxoxoxxxxoxoxxxxxxxxoxoxxooooxxxxxxxxxxxx
```

To make it easier, a vehicle will not be split across different lines of data

Input: There are 10 lines of data, each 50 characters long.

Output: Show the number of each type of vehicle each on a separate line.

Example Input

```
xooxxxxooxxxxoxoxxxxoxoxxxxxxxxoxoxxooooxxxxxxxxxxxx
ooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
oxoxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
oxoxxoooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxoxoxxooooxxxxxxoxoxxoooxxxxxxxxoxoxxoooxxxxxxxxx
xoxoxxoooxxxxxooxxxooxxooxxxxxxooxxxxooxxxxooxx
oxoxxoxoxxoxoxxoxoxxoxoxxxxxxxoxoxxxxxoxoxxxxxoxox
xooxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxoo
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxoxoxxooo
```

Output to screen:

```
12 small
11 medium
7 large
```

This page intentionally left blank

C. Reverse GREEN

When using strings there is usually a function called substring. Your task is to do the opposite of the substring function. Rather than returning a specified substring within some string, you will output the given string with the substring taken out.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will be one line with a string and two integers, separated by spaces. The first integer is the start index of the substring to be taken out, and the second integer is the end index.

Output

Output the given string, with the substring taken out specified by the given integers. The output will be `n` lines, with no leading or trailing white space.

Example Input

```
3
COMPUTER 1 3
SCIENCE 3 7
RULES 3 4
```

Example Output to Screen

```
CPUTER
SCI
RULS
```

This page intentionally left blank

D. Square ORANGE

Your mother has asked you to create a template for the squares of a quilt she is making. Her quilt will be based on words! Print out a square of the given word, in the style shown in the example output.

Input
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a single line with one word of length `k` (`0 < k < 100`).

Output
Output the square in the format shown in the example output. There are no spaces between data sets. The final square should be both as wide and as tall as the length of the word.

Example Input
```
3
one
three
fifteen
```

Example Output to Screen
```
one
n n
eno
three
h   e
r   r
e   h
eerht
fifteen
i     e
f     e
t     t
e     f
e     i
neetfif
```

This page intentionally left blank

E. Robbery LIGHT BLUE

The newest video game just came out. However, you are short on money. You have decided to turn criminal and rob a bank with a squirt gun. Ignoring the reliability of your plan, you need a program that calculates the most value you can take from the bank given the amount of weight you can hold. Because you are only so strong (and can only carry so much weight), you need to find the optimal combination of items to take with you to get the most payoff.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with two integers on a line, `W` and `O` respectively. `W` represents the maximum amount of weight that you can hold. `O` represents the number of items in the bank. On the next line, there will `O` integers that represent the value of each item. On the following line, there will be `O` integers that represent the weight of the object directly above it on the previous line. Objects are finite and can only be used once.

Output

For each data set, output the maximum value you can steal from the bank on a single line.

Example Input

```
2
10 7
1 2 3 4 5 5 5
1 1 1 1 1 1 1
5 5
5 10 15 20 5
2 3 2 3 5
```

Example Output to Screen

```
25
35
```

This page intentionally left blank

F. Holes DARK BLUE

After receiving an impenetrable box, you wonder what's inside! Write a program that finds out how many different disconnected sections the box has, and the total area of the space within the box.

Input
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with two integers `r` and `c` representing the number of rows and columns of the box, respectively. The next `r` lines will represent the box, with `#` representing walls and `.` representing spaces. The box will be surrounded entirely by walls.

Output
Output the number of discrete (disconnected) sections and the total number of spaces in the box, in the format shown in the example output.

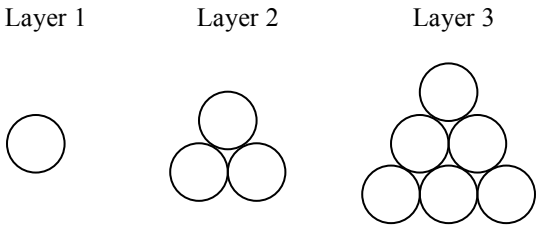
Example Input
```
2
4 8
########
#...#..#
#.#.#..#
########
3 3
###
#.#
###
```

Example Output to Screen
```
2 sections, 9 spaces
1 section, 1 space
```

This page intentionally left blank

G. Ornaments RED

As a worker at the local mall, you have to set out the Christmas ornaments display. The display setup is a triangular pyramid, with the top layer containing one ball, the second from the top containing three, and so on. Each layer's side length will be equal to their layer number. Write a program to determine the total number of ornaments in a pyramid, given the number of layers.



Input
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a single line containing one integer, denoting the number of layers in the pyramid.

Output
Output the total number of ornaments in the pyramid.

Example Input
```
2
1
4
```

Example Output to Screen
```
1
20
```

This page intentionally left blank

H. Family LIGHT PURPLE

You have been given bits and pieces of your family tree. Your task is to determine if two people are related based on several connections.

Input
The first line will contain a single integer `n` that indicates the number of connections. The next `n` lines will consist of a name, a connection, and another name. The connections will be either `mother`, `father`, `brother`, `sister`, `daughter`, or `son`. The next line will contain a single integer `m` that indicates the number of test cases. The next `m` lines will consist of two names. Your program should determine if the two names are related.

Output
Output either `Related` or `Not Related`, depending on whether they are connected or not. There will be `m` lines of output.

Example Input
```
3
John brother Susan
Kim mom John
Dave son Jim
2
Jim John
Kim Susan
```

Example Output to Screen
```
Not Related
Related
```

This page intentionally left blank

I. Checkpoint DARK PURPLE

A drone is being tested by finding the shortest path through a maze, reaching every checkpoint on the way in order. Your task is to write a program that finds the shortest path from the start, through every checkpoint in order, and then to the exit, then prints the length of that path.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with three integers `r`, `c`, and `d` representing the number of rows and columns of the maze and the number of checkpoints, respectively. The next `r` lines will make up the maze, with `S` being the starting point, `E` being the end point, the numbers `1-9` being checkpoints, `#` being a wall, and `.` being an open space. `S` and `E` also count as open spaces.

Output

The output will be the length of the shortest path from the start, through every checkpoint in order, and to the exit. There will be `n` lines of output with no trailing whitespace.

Example Input

```
2
5 8 2
S....1..
.######.
..2.#...
.######.
......E.
1 11 2
S...1...E.2
```

Example Output to Screen

```
24
12
```

This page intentionally left blank

J. Bags BLACK

When creating a gift bag, you are trying to create a bag that weighs as much as possible with as few items as possible. This is to make it feel like the guests are receiving a lot without having to use as many items in each gift bag. Instead of finding how much to put in each gift bag, write a program that finds the fewest number of items you can put into a gift bag to reach the recommended value.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will be three lines, and will start with a single integer `x` denoting the number of items. The next line will contain `x` integers, indicating the weight of each item. The next line will be a single integer indicating the total weight you are trying to reach.

Output

Output the smallest number of items that will add up to the weight to be returned. If it is not possible to add up exactly to the weight to be returned, print `Not possible`.

Example Input

```
3
10
1 3 3 3 5 7 7 5 5 10
39
10
1 2 3 4 5 6 7 8 9 10
27
1
100
50
```

Example Output to Screen

```
6
3
Not possible
```

This page intentionally left blank

K. Bomb WHITE

You decide to create a game involving a 3d maze with destructible walls, where all the character has to work with is bombs. In order to determine the number of bombs to provide for each level, you need to know the minimum amount necessary to reach the exit and base it off of that. Your task is to write a program that will find the smallest number of bombs necessary to reach the exit. Each bomb can destroy one wall, leaving a blank space in its place.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with three integers `f`, `r`, and `c`, representing the number of layers, rows, and columns, respectively. The next `f` sets of `r` lines will be the maze, with every set of `r` lines being one layer of the maze.

Output

Output the smallest number of bombs necessary to escape the maze. There will be no trailing white space.

Example Input

```
2
2 3 3
S##
##E
###
#.#
#..
###
1 2 10
S#.####.#E
..##..###.
```

Example Output to Screen

```
1
5
```

This page intentionally left blank

L. Sudoku SILVER

You are tasked by your very smart uncle to solve sudoku puzzles. However, you have never actually done sudoku. Your uncle teaches you that sudoku is a grid based number puzzle where no value is repeated on the same column or row. The values used in the puzzle range from 1 to 9. Furthermore, your uncle explains that the grid of numbers has 9 larger squares. Each of the 9 squares contain 9 numbers from the grid. The squares can be formed by drawing straight horizontal and vertical lines every 3 numbers across and down the grid. You uncle explains that a number can only be used once inside of each of the 9 squares. After informing you, you uncle challenges you to solve a puzzle. Your job is to write a program to solve the sudoku puzzle.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a 9x9 grid of numbers. This grid of numbers represents the sudoku puzzle. All blank spaces in the puzzle are represented by 0s. All numbers (1-9) represent parts of the puzzle that have already been completed. Your job is to complete the puzzle by replacing the 0s with the correct numbers. There will be a blank line between each grid of numbers.

Output

For each data set, output a 9x9 grid of numbers that represents the completed sudoku puzzle. After each data set, **EXCEPT** for the last, print a blank line.

Example Input

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

Example Output to Screen

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

This page intentionally left blank