Problem Set

Trinity University ACM
High School Programming Competition

April 26th, 2014

Problem 0 - Pacman

While playing Pacman in the arcade one day, you're getting wrecked by the ghosts and all your friends are making fun of you. You want to show them who's the boss of the arcade. In the game, Pacman walks through a maze eating pellets, but some pellets will give Pacman the ability to eat the ghosts. You've figured out how to line up all the ghosts in front and behind Pacman when you get the special pellet, but you only have a limited amount of time to eat them in any order! Once you eat the pellet, the ghosts walk 1 feet per second in a line away from Pacman while Pacman can run 2 feet per second to try to catch them. The movement of Pacman and the ghosts happen simultaneously. This means that if there are two ghosts, 1 feet and 2 feet away from Pacman, and Pacman runs towards the ghosts, he will catch the first ghost 2 feet away from Pacman's original position. By then, the second ghost would have reached 3 feet from Pacman's original position.

Input:

The first integer of the input will contain the number of data sets. Within each data set, there will be two lines. The first line will have a single integer that represents how many seconds Pacman has the ability to eat ghosts. The next line will have four integers. Each integer will be the distance of the ghosts from Pacman in one line. Positive integers mean that the ghosts are in front of Pacman and negative integers mean that the ghosts are behind Pacman. Pacman can change direction instantly and without penalty. All units of time are in seconds, units of distance are in feet, and Pacman can walk a maximum of 2 feet per second.

Output:

For each set of data, display "Success!" if it is possible for Pacman to eat all the ghosts within the time frame. Display "Failure" if it is not possible for Pacman. If Pacman reaches the last ghost at exactly the end of the time limit it counts as being caught.

Sample Input:

```
3
15
1 2 3 15
20
12 -2 4 6
18
-6 -5 18 7
```

Sample Output:

```
Success!
Success!
Failure
```

Problem 1 - Tetris

Description:

You are training a computer program to be the best Tetris player ever. In this version of Tetris, you are given a 10 by 10 grid to play on in which different shaped blocks, called Tetriminos, fall into the playing field. The objective of the game is to manipulate the Tetriminos, by moving each one sideways and rotating it by 90 degree units, with the aim of creating a horizontal line of ten blocks without gaps. When such a line is created, it is cleared from the game, and any block above the deleted line will fall. Points are given based upon the number of lines cleared at once, one point per line. The game ends when the stack of Tetriminos reaches the top of the playing field and no new Tetriminos are able to enter, however in this problem you will not be asked to analyze if the game is over nor will you be given a playing board in which a piece cannot fit.

In your efforts to teach your computer how to play Tetris, you decide to isolate certain conditions. One condition will be checking to see if the the next given block can clear any rows, because this will often be the best move for the computer to make. Your goal is to check for this condition and determine how many lines can be cleared with the given Tetrimino. All placements are allowed as long as they don't overlap with existing pieces, even if there is no way to move the piece to that location from the top of the board.

Input:

First line contains T (T<100), the number of test cases.

Each input begins with the board represented by a ten by ten grid with 8's representing current Tetriminos and 0's representing empty space. That is followed by the next Tetriminos piece in a four by four grid, 8's representing the Tetrimino piece and 0's representing empty space surrounding the Tetrimino.

No full lines will appear in the original input.

Output:

For each test case, output the number of lines the Tetrimino piece can clear.

Sample Input:

```
4
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000888008
```

0008888888
0000
0888
0080
0000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0888888888
0888888888
0000
8888
0000
0000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000880800
0000880800
0008888800
0888888888
0000
0880
0880
0000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000888800
8888880088
8888880088

0000
0880
0880
0000

Sample Output:

1
2
0
2

Problem 2 - Watch Your Wallet

Description: It's Spring time and your third-uncle-twice-removed has decided to present you with a Steam gift card of $100. With Spring Steam sales going on, you can get a lot for your buck. You want to get the maximum amount of playtime. There are various discounted games.

Input: The first line is the number of test sets. The second line denotes the number of games, G<=20, in that test set. For every input, each game is a 3-tuple (game name, playtime, price). The playtime and price will be integers.

Output: For each input you will output a single number with the maximum total playtime you could buy for your $100.

Example Input:

```
2
6
Left 4 Dead, 20, 15
Rust, 15, 20
Sleeping Dogs, 25, 15
Final Fantasy 7, 35, 14
Bioshock Infinite, 30, 60
Dishonored, 20, 45
4
Heracles, 1, 50
Reid's Bacon, 45, 25
Doctor How, 34, 15
Hotdog Carnival, 24, 20
```

Example Output:

```
100
103
```

Problem 3 - Mario Galaxy

Mario has fallen off his face ship and landed somewhere in Star Road. He knows that if he can get to a powerful enough warp star he can return, but all he can see are lesser stars that will not get him far. Determine if he can escape Star Road and the fewest number of steps it will take him.

Input: The first line of the input will be how many input sets there are. Each input sets will begin with a line consisting of the number of columns, C<50, followed by the number of rows, R<50. Next will be R lines giving the map with 1s indicating walls, 0s indicating navigable space, lowercase alphabet characters indicating a connecting warp to the matching letter, a starting point of capital S, and a goal of capital E. Consider the warp letters to be the same points in space, stepping onto one c lets you step off of the other without counting a step from c to c. It is possible to pass over a warp without using it. Warps can only have 2 entrances/exits. Every board will include a start and an end.

Output: if it is possible to reach the goal, return YES followed by the minimum number of steps needed. If not, print NO.

Sample Input:

```
1
24 10
111111111111110f1111111
101110b001111e0011111111
1S1111100c1110111f000111
101b111111111111011E111
100a111c00d1111110111111
111111101101111111111111
0a111110d001111111111111
10111111e111111111111111
111111111111111111111111
111111111111111111111111
```

Sample Output: YES 21

Problem 4 - Mario Party

Description:

You are over at a friend's house for a slumber party. The activities during the night have been a lot of fun but they weren't going exactly as planned. Along with feeling fat due to a large consumption of food and after being repeatedly beaten in front of your friend who you have a crush on, you are currently not in the best of moods. You land on a duel spot (1 vs 1) on the Mario Party game board. It randomly selects one of the other players and it picks your best friend, Mark. You know that Mark has done nothing to you; in fact, Mark has been trying hard to get your crush to pay attention to you - as good as any wingman a you could ask for. The game you have to play revolves around tiles being infected with a koopa virus. The koopa virus will infect one tile at a time. This tile will move up one "infection" level until it reaches the maximum infection level. At that point, that tile will infect adjacent tiles (north, south, east, west).

Figure out what effect the virus Mark is spreading will have on your land. This will somehow help you get the girl and win life.

Input:

The first number of the input represents the number of data sets ($0 < a < 100$). The next line will be the number of rows and columns of the grid for the particular data set ($0 < b < 20$, $0 < c < 20$). The following b lines will contain the state of the grid at the start (the state of the various tiles will be represented with an **A**, **B**, **C**, **D** or **X**. It will be made up of symbols that represent the state of each location. This will be followed by the number of events ($0 < d < 20$). The last line(s) of each data set will be the row and column of each location ($0 < e < 20$) akin to a coordinate where an event occurs.

Each event coordinate composes of two number: the first number is the row and the second number is the column. For example: the event occurring at **1 0** will be on the **A** tile and the event occurring at **0 2** will be on the **X** tile.

B C X
A D B

A tile that has **A**, **B** or **C** will go up one state (**A**->**B**, **B**->**C**, **C**->**D**) during an event. A tile in the **D** state will not change states but instead will fire an event in adjacent tiles (north, south, west, east). A tile in the **D** state will only spread the infection once per event. (It doesn't jump back and forth between adjacent D tiles.). A tile that has **X** means it cannot be infected by the virus and will not infect another.

Output:

The output should be composed of the final state of the grid, after all the events have occurred.

Sample Input:

3
2 2
AB

CA
2
0 1
1 1
4 4
AABC
XAAB
XXAD
DABB
3
0 1
2 2
3 3
2 3
BAC
XDD
1
1 1

Sample Output:

AC
CB
ABBC
XAAB
XXBD
DABC
BBD
XDD

Problem 5 - Asteroid Face Off

Description:

Captain! We're on a direct collision course with an oncoming asteroid! There's no time to avoid it, we must blast our way through! Tensions are high but being the intelligent programmer you are, you'd rather know if you're about to die or not.

Given the time till impact, your ship's fire rate, and the size of the asteroid, will you survive? Every hit on the asteroid reduces its size by one. Your first shot happens at the end of a delay. If you get a shot off at the instant the asteroid would hit you then you survive.

Input:

Input will consist first of a line indicating the number of problem sets that will follow. For each set there will be a single line containing 3 positive integers, the time until impact, your ship's fire rate(the time between shots), and the size of the asteroid.

Output:

For each set, print whether or not you and your crew will survive. If you should survive, print "We made it!" otherwise print "Splat!".

Sample Input:

4
3 1 2
20 5 5
2 1 2
10 3 4

Sample Output:

We made it!
Splat!
We made it!
Splat!

Problem 6 - Princess Peach

Description:

Princess Peach has once again, unsurprisingly, been captured by Bowser. However, she has found a way to finally escape from Bowser's castle herself instead of waiting around for Mario. It is Bowser's birthday and in a fit of chauvinism in which he assumed that all women know how to bake Bowser has demanded that Princess Peach make him a birthday cake. Unfortunately, Peach reinforced Bowser's masculine ignorance by being excellent at baking. She made her way to the kitchen and while exploring found some leftover sleeping powder in a spiderweb filled cupboard. Peach knows that she can use the powder in the cake to put Bowser to sleep and facilitate her escape, but she needs to get the ratio of sleeping powder in the cake exactly right. If she puts in too much, Bowser might taste it before he falls asleep and lock her up where Mario will never be able to find her again. But if she doesn't put enough, then Bowser will wake up and catch her before she can run far enough away and then lock her up permanently, causing the chain of Mario games to come to an end.

For each recipe you need to find the proportion of sleeping powder in the cake as a fraction of the total contents. If the cake is composed of less than 1/10th sleeping powder then Bowser will catch Peach when she tries to run away. But if the cake is more than ¼th sleeping powder then Bowser will taste it and get ridiculously, therapy-worthy mad.

Input - The input will first consist of an int, R (1 <= R <= 50), that represents the number of recipe sets. Then each set will have four non-negative ints that are the amount of sleeping powder, flour, sugar, and eggs in that order.

Output - For each input set print either "NOT ENOUGH", "TOO MUCH" or "JUST RIGHT".

Sample Input:

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

Sample Output:

```
NOT ENOUGH
TOO MUCH
JUST RIGHT
```

Problem 7 - Bit Shifter

Description:

You are playing an intense game of Civ 5. You are currently at war against your two friends, Lewis and Myers. They seems to be communicating purely by encoded messages. You happen to receive a message by accident. Each one sends a single integer value. You manage to figure out that the message is encoded in the binary. Of course! How else would they correspond with each other… After watching Lewis and Myers sack a couple of countries, you start to crack their code. You must now be prepared if they decide to attack you. Use what you have discovered to figure out which country they will attack, how they will attack it (by land or by sea), and how strongly.

Lewis made his message harder to interpret by inverting (replace 1s with 0s and 0s with 1s) the rightmost four bits in his private message when he sends it. So you get the public version with the bits flipped. For example, if his original value was 00000001 then you get 00001110. Myers' third to last bit will decide by land or by sea. If it is a 1 they will attack by sea, 0 for land (0100 would be sea, 1011 would be land). To get the power of the attack you have to upshift (left-shift) Myers' code four bits and combine it with Lewis' private code, where any bit that is on in either string is on in the result. The last three bits of the power contain the integer number for which country (0-7) will be attacked.

Input - The first line will be the number of inputs. In each input, the first line will be Myers and the second line will be Lewis.

Output - For each input you write a single line with the format "Country {country} by {land/sea} with power {power}", where the names in curly braces are replaced by the appropriate values.

Sample Input:

3
2
1
10
3
87
7

00001 11110

Sample Output:

Country 6 by land with power 46
Country 4 by land with power 172
Country 0 by sea with power 1400

Problem 8 - Dragon's Lair

Description - It is 1984, and you are playing the new Dragon's Lair video game at your local arcade. Suddenly the machine breaks and can't determine whether or not a user is winning. It's your job to quickly write a program that can determine, based on the users' inputs through a joystick, whether or not they will win the game.

Input:
The first line of the input will contain an integer g, that represents the number of times that the game will be played. Each time the game is played, the first line will contain an integer n<100, representing the number of trials that your hero must face in this game. There will then be n lines containing sentence long descriptions of the trials that your hero will face (ending with a period or exclamation mark), three spaces and then the required input to defeat this trial. The description will not contain sequences of three consecutive spaces. After this, there will be n lines containing the user inputs during the game.

Output:
The output should contain n lines, where each line holds a string representing the result of the game. If the user input for a particular game, matches the required input to beat all of the trials in that particular game, then output "You Win!". Otherwise, output "You Died!".

Example Input:
2
3
You see a dragon. up
The castle's drawbridge is raising. left
The walls are closing. down
up
left
down
2
Ninja Attack. right
Lava! right
right
left

Example Output:
You Win!
You Died!

Problem 9 - Pole Position

Description: You are playing the classic racing arcade game “Pole Position”, and you are concerned that you may not be able to complete the track within the time limit. As you may recall, there are several checkpoints that you must pass within specific time limits, or else the game ends. When you reach a checkpoint, you are awarded more time to reach the next checkpoint, carrying the time you had left with you. Your car is magical, and goes at a constant speed of 120 feet per second. While racing, you decide to write a program to determine how many checkpoints you can complete. If you hit a checkpoint at the exact time your time expires you pass that checkpoint.

Input: You will be given a line containing one integer, $1 \leq n \leq 100$, indicating the number of test cases. Each test case starts with a line containing two positive integers, the first indicating the number of checkpoints in the track ($1 \leq m \leq 50$), and the second indicating the amount of time you have when the race starts. There will then be m lines, each containing two space-separated positive integers. The first integer indicates the distance to the checkpoint in feet, and the second integer indicates the time awarded by reaching the checkpoint.

Output:
Print the number of checkpoints that the car can complete before failing to reach a checkpoints.

Sample Input:
```
2
2 20
1800 20
2400 0
4 30
3000 5
600 0
2400 100
10 0
```

Sample Output:
```
2
2
```

Problem 10 - DDR Synchronization

Description:
DDR - You are playing Dance Dance Revolution with a friend. The song you are dancing to requires you to work together to beat the song. You need to coordinate your moves with Player One.
Read in Player One's move, and according to rules/patterns, return your moves.

Player One's moves will be a string four digits long consisting of `1`'s and `0`'s. The digits from left to right represent the current state of the dance pads left, up, down, right respectively. A `1` means the pad will be pressed at that time. You only have two legs, and therefore may only tap two pads at once.

When player one taps the left dance pad, you should tap on the right dance pad.
When player one taps the right dance pad, you should tap on the left dance pad.
When player one taps the up dance pad, you should tap on the down dance pad.
When player one taps the down dance pad, you should tap on the up dance pad.

Input Description:
The first number represents the number of songs. The first line of each song is the number of dance moves, **n**, in that song. The next **n** lines will be player one's set of moves.

Output Description:
For each move you output the proper footing for the right partner. Put a blank line between dances.

Sample Input:
```
2
5
0001
0100
0101
1001
1000
3
0010
1100
1001
```

Sample Output:
```
1000
0010
```

1010
1001
0001

0100
0011
1001

Problem 11 - Crafting

You are a great minecraftsman in the land of Minerim. You are tasked with looking at the market values of items that you can craft and then deciding, based on your inventory of raw materials, what items to craft in order to maximize your profit.

Input:

The first line in the problem set will contain an integer that represents the number of input sets that you will be considering. The first line of each input set will contain a comma separated list of the items in your inventory with their quantity and name. This is followed by a line with a single integer, R, for the number of “crafting recipies”. The next R lines will contain the name, the crafting recipe amounts and ingredients, and the value of an item that can be sold to market. Raw materials appear in this list with the string “Raw Material” in place of the ingredients to give you their market value. There will be no more than 60 items in the initial inventory and all ingredients will be listed as the product of a recipe earlier in the list. All materials in the problem, including all raw materials, will have an associated recipe. There will be no more than 8 recipes that are not simple raw materials.

Output:

Each line of the output set should contain an integer representing the maximum possible amount of money that you can make from selling crafted items from a particular input set, given your starting set of raw materials.

Sample Input:

```
2
4 Corundum Ore, 10 Iron Ore
3
Corundum Ore - Raw Material - 20 Gold
Iron Ore - Raw Material - 2 Gold
Steel Ingot - 1 Iron Ore, 1 Corundum Ore - 20 Gold
3 Leather, 6 Iron Ore
7
Animal Pelt - Raw Material - 5 Gold
Leather - 1 Animal Pelt - 10 Gold
Iron Ore - Raw Material  - 2 Gold
Iron Ingot - 1 Iron Ore - 7 Gold
Leather Strips - 1 Leather - 9 Gold
Iron Chest Plate - 1 Leather Strips, 5 Iron Ingot - 125 Gold
Hide Bracers - 1 Leather, 1 Leather Strips - 10 Gold
```

Sample Output:

```
100
152
```