Problem Set

Trinity University ACM
High School Programming Competition
April 5th, 2008

Problem 0 - The Evil Doctor Plays Chess

An evil doctor has kidnapped all the queens in all the nearby kingdoms. Since it is your duty as the knight in shining armor to rescue the queens, he has brought you to his evil lair to play a game. The evil doctor is a big big fan of the game chess. However he only has queens at his disposal so he's changed the rules a bit. The evil doctor will have the kidnapped queens arranged on a giant board game. It is your job to determine whether or not any of the queens are in danger of being attacked by any of the other queens using normal chess rules. If you get it right enough times he'll set the queens free, if not.... dun dun duuuunnnnn. Since you're not much of a knight in shining armor and more of a computer scientist, write a program to help you out.

Input: Input will begin with a line containing an integer m representing the number of datasets. Each data set will begin with a line containing two integers x y where x represents the width of the board game and y represents the length of the board game. The next y lines will contain x characters. Queens will be represented as 'Q' and empty spaces will be represented as '-'.

Only the characters 'Q' and '-' will be used.

Output: For each input set you will print one word. If you find a conflict printout “Conflict”. If you don't find a conflict, printout “Safe”

Sample Input:

```
2
4 4
---Q
--Q-
-Q--
Q---
4 4
-Q--
---Q
Q---
--Q-
```

Sample Output:

Conflict
Safe

Problem 1 - Spy vs Spy

You are a cryptanalyst in Julius Caesar's legandary thirteenth legion during a turning point in the war against Pompey Magnus. Caesar has entrusted you to find out all the information you can from encrypted messages that were intercepted by his scouts. These messages were being sent from Pompey to the senators in Rome. Lucky for you, Caesar invented the only method used to encrypt messages at this time, the Caesarian Cipher.

The Caesarian Cipher works by picking a number n where 0 <= n < 26. Then each letter in the message is switched with the letter that falls n characters away in the alphabet. For example if n is 3, the message abc would become def. The system works modularly so the message xyz with an n of 4 would become bcd.

As a cryptanalyst, you have a few tricks up your sleeve when it comes to cracking codes. You decided to pick a few keywords, and then check to see if any of the encrypted words could be one of your keywords.

Input: Each data set will begin with two integers `m s`, where m indicates the number of lines of cipher text and s indicates the number of lines of keywords.
The next `m` lines of cipher text will contain one encoded word (whose length is greater than 1).
The next `s` lines of keywords will contain a single word (whose length is greater than 1).

Output: For each encoded word, output a line containing the keyword it could be and the integer key n separated by a space. If the encoded word could be multiple keywords, order them in the same order they appeared in the input.

Sample Input:
3 4
jason
bbbbb
zzzzz
pgyut
jason
bbbbb
zzzzz

Sample Output:
jason:
pgyut 6
jason 0
bbbbb:
bbbbb 0
zzzzz 24
zzzzz:
bbbbb 2
zzzzz 0

Problem 2 - Dr. XKCD

The evil Dr. XKCD has walked into a restaurant with a certain amount of money. He threatens to blow up the city unless you provide him with an optimal way to spend so that he is left with exactly $0.

Input: The input will begin with a line with a single integer, n, representing the number of datasets to evaluate. Each dataset will begin with a line containing two integers, m and c, where m is the amount of money the evil doctor has in cents, and c is the number of items on the menu (c<10). The next c lines, representing the items, will consist of an item name (single word) and a price (also in cents).

Output: For each dataset, you will need to produce one of two possible outputs. If you are unable to fulfill the the doctor's request, print out "Quick, change the price of something!" If it is possible to produce an order using all of Dr. XKCD's money, print out "You should order the following:" followed by a list of the items he should order. For each item, print out the number of that item necessary to complete the order and the name of the item. If an item is not used in the solution, do not print it out. There will never be two correct solutions.

Sample Input:
2
500 4
fries 99
coke 199
hamburger 399
nuggets 299
1500 6
fruit 210
fries 275
salad 335
wings 355
cheese 420
sampler 580

Sample Output:
Quick, change the price of something!
You should order the following:
1 fruit
2 wings
1 sampler

Problem 3 - Gauntlet

You have recently begun playing a Massively Multiplayer Online Role-playing Game (MMORPG) because your friends talked you into playing with them. In the game, you are given a quest by an elderly gentleman in town: you must kill several monsters in quick succession. The speed at which the battles begin hinders any possible health regeneration.

Luckily, before embarking on this task, you searched for the quest online, and found a guide detailing how the monsters act and their Damage Per Second (DPS). The programmers who designed these monsters had no concept of artificial intelligence and therefore each monster attacks the person in your party with the largest amount of health, and it continues attacking that person until it is dead. Once that person is dead, it attacks the next person with the largest amount of health. If two party members have the same health, then it chooses the one with the highest DPS. Using this information and each party member's DPS, you decide to write a program to determine if your party can kill all of the monsters before everyone is dead. Due to some inexplicable reason, no two party members will have the same DPS.

Death means that a person's health is less than 1. Assume that damage is dealt on a per second basis, and that all attacks (from both party members and the monsters) happen every second and at the same time. So if the monster would die from one of the party member's attacks, the monster still does his damage for that second. Similarly, if a character dies in a given second that character's damage is still done to the monster. Characters who are killed do not do damage in subsequent seconds.

Input: You will be given an integer n on a single line that will be represent the number of gauntlets to run (0<n<50). Each gauntlet is presented as 5 lines of character data, an integer, m, which is the number of mobs, and then m lines representing each mob. For each line of character data, you are given 2 integers, health and DPS. For each mob, you will also be given 2 integers, health and DPS.

Output: For each gauntlet, you will determine if your party survives; that is to say, at least one party member is alive after the last mob has died. If your party survives, you will output "Gauntlet n : Survived", or if all of your party members die, you will output "Gauntlet n : Failed" on a line, where n is the number of the current gauntlet being run.

Sample Input:

```
2
100 2
50 10
20 30
40 5
30 6
3
60 10
50 60
10 100
10 5
12 3
```

8 4
16 8
11 2
2
100 5
200 20

Sample Output:
Gauntlet 1: Survived
Gauntlet 2: Failed

Problem 4 – Poor Salesman

You are a low level door-to-door salesperson, and thus do not service a large area. Your supervisor asks you to pick three cities from a list to be your region. Being the frugal and smart person that you are, you decide to write a program to find the group of three cities with the shortest combined distance between them (shortest perimeter for the triangle they form). Your boss has provided you with the locations of the cities so you write a program that will help you save money on gas.

Input: The input will begin with a number telling you how many input sets you have to consider. Each input will begin with the number of cities to be considered, N<27. For simplicity we will assume that the cities are named 'A' through 'Z'. After the number of cities will be N lines giving the distances (all integers) from the *i*th city to all the other cities. This will be a symmetric NxN matrix with zeros on the diagonal.

Output: For each input set you will output a single line. That line needs to have the names of the three cities that have the smallest triangle printed in alphabetical order as capital letters separated by spaces.

Sample Input:

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

Sample Output:

```
A B D
B C D
```

Problem 5 - Pizza

Billy has been given the task of finding the best place for us to buy pizza for the competition. It's really a fairly simple task because all we care about is cost. The cheapest place is the best. So Billy has called a bunch of places and asked them how much a medium pizza costs. All that needs to happen now is to figure out which one is the cheapest. Your program is going to do that.

Input: The input will begin with a line that has a single number for how many inputs you should process. Each input will begin with the number, $0<N<100$, of places that Billy called. That will be followed by N lines where each line has the name of the place as a single word followed by the price of a medium pizza at that place. The prices do not have to be integers (though they could be) and no two prices will ever be equal.

Output: For each input you just want to print out the name of the place with the lowest price.

Sample Input:

```
2
3
Hut 7.95
Ceasars 4.95
Dominoes 5.95
4
a 25
b 89
c 1
d 74.99999
```

Sample Output:

```
Ceasars
c
```

Problem 6 – Gambling Glory!

A fund raiser for helping teens addicted to MMORPGs (such as WoW and Evercrack) is going on and you've decided to participate. To raise money they have invented an odd game. Everyone pays $20 to join and their name goes on a wheel. Then they guess when their name will be picked. For example, if they pick 5 and their name is the 5th name picked they win a share of the pool. After everyone has paid up and their name is on the wheel the game begins.

The game begins with the arrow pointing at the first location. They spin the wheel and remove the name that the arrow lands on. The arrow then points at the next name and to prevent the wheel from landing on an empty space everything is spaced evenly after a name is removed. The game continues until there are no names left. For example, if there are four names on the wheel: "Bob", "Alice", "Tim" and "You", and the first spin goes forward two spaces. "Tim" would be removed and if he picked 1 he would win. The wheel now contains "Bob", "Alice and "You" with the arrow pointing at "You". Say the next spin moves another two spaces, it would then land back on "Alice". The third spin would hit “You” and if you had guessed that you would be the third name selected you would win.

After a couple games you notice that the wheel always moves forward the same number of spaces. Knowing, n, the number of people playing, p, your original position in the wheel before the game begins, and m, the number of spaces the wheel moves each spin, write a program that will tell you when your name will be selected.

Input: The first line will contain a number representing the number of data sets. There will be one line for each data set containing three integers each separated by a space. The first integer $1<=n<2^{29}$, will be the number of people playing. The second one is $1<=p<=n$ and it will will represent your original position on the wheel. The last one, $m<2^{29}$ is how many spaces the wheel moves each spin. The count for the position begins with 1.

Output: For each data set, output the integer representing the number of spins it takes for your name to be selected. The first spin is spin one.

Sample Input:

```
3
5 3 7
6 2 3
1000 3 7
```

Sample Output:

```
1
2
910
```

Problem 7 - Dr. Mem O. Ize

The evil Dr. Mem O. Ize has developed a weapon of mass destruction that will destroy the planet. Of course, like all proper mad scientists he has put a disable mechanism on the device. The mechanism appears to be a simple pegs game where the player "jumps" pegs in an attempt to finish with as few pegs as possible. To deactivate the device, you must have only one peg remaining.

You have just run into the room with the deactivation device to find your sidekick playing the game. At this point fear grips you because your sidekick isn't the brightest star in the sky. Pushing him out of the way you analyze the board to determine if wonder boy has doomed the fate of humanity or if everyone can be saved. The board is a 5x4 rectangle with markers for where there are “pegs”. Jumps are allowed in the horizontal and vertical direction, but not diagonally. When a jump is done the jumped peg is removed from the board. For this problem you need only determine whether it is possible to disable weapon.

Input: The input will begin with a line giving you the number of boards that you need to consider. Each input will consist of four lines each with five characters on it. An 'O' represents a peg, and a '.' represents and empty slot.

Output: For each board you will print out one line. If it is possible to play the game down to one peg you print out, “Let's disable this thing.” Otherwise you will print out, “Boom!”

Sample Input:

```
3
O....
.....
.....
....O
OO...
O.O..
.O...
.....
OOOOO
.OOO.
.O.O.
OOOOO
```

Sample Output:

Boom!
Let's disable this thing.
Let's disable this thing.

Problem 8 - How Many Shirts?

In the process of planning the programming competition we have to decide how many T-shirts to purchase. One would think that the ideal way to do this is to buy exactly enough shirts, but with the way that T-shirt companies price things that isn't always the case. Of course, you have to buy enough shirts for everyone in the competition to get one, but in some cases it actually costs less to buy more. For this problem you need to tell us how many shirts we should purchase given the number of contestants and a pricing schedule.

Input: The input for this problem will begin with the number of input sets you have to consider. Each input set will have the following format. On one line you will have the number of contestants. The line below that will be the number of pricing options followed by that many lines for the pricing options. Each pricing option line will have two numbers. The first is a price and the second is the minimum number of shirts they will give you at that price. The first price listed will always be for 0 shirts.

Output: For each input set you simply print out the number of shirts that you should buy so that you will have at least one for each person yet minimize the total cost.

Sample Input:
2
40
2
12.75 0
10.00 30
60
3
15.00 0
10.25 36
8.25 72

Sample Output:
40
72

Problem 9 - Can we date?

It has been posited (in XKCD) that it isn't creepy to date someone unless they are younger than the following formula:

$$\left(\frac{age}{2}+7\right) .$$

Using this formula you need to decide if certain people are safe to date.

Input: The input will begin with a single line giving the number of input sets you are supposed to consider. Each input set will have a single line with two numbers on it: your age and the age of the person you are considering dating.

Output: The output will have one line for each input set. It will say, "That's a bit creepy." if the first age is too much bigger than the second age. Otherwise it will say, "Go for it!"

Sample Input:
4
45 35
16 12
99 35
34 24

Sample Output:
Go for it!
That's a bit creepy.
That's a bit creepy.
Go for it!