Problem Set

Trinity University ACM
High School Programming Competition
April 8th, 2006

Problem 0 - 2nd Grade Homework

(Don't all good things start counting as 0?)

A common assignment for early grade school spelling is to alphabetize the words. Every child seems to hate this assignment and let's be honest, does it really help you learn how to spell the words? To help counter the efforts of cruel teachers across the nation you decide to write a program to foil their nefarious plans. You write a program that will allow students to type in their spelling words and it will print them back out in sorted order.

Input: The input for this problem consists of a header line that tells you how many words are on the spelling list for that week. It will be followed by that many lines with one word per line.

Output: For the output of this problem you should print the words in alphabetical order, one word per line.

Sample Input:

```
5
cot
dot
dog
cat
ape
```

Sample Output:

```
ape
cat
cot
dog
dot
```

Problem 1 – Bell Curves

In college it is not uncommon to have professors who grade everything on a curve. In situations where the highest grade on an exam is a 57, this isn't such a bad thing. Granted, some people don't like the fact that half the class is certain to get a C or worse. That's college for you though.

The way grading on a bell curve works is that the mean grade is the line between a B and a C. Since we assume you have a "nice" professor, anyone landing exactly on a dividing line gets the higher grade. Breaks between other grades happen at spacings of standard deviations. So if you were one or more standard deviations above the mean you get an A. If you were more than one standard deviation below the mean, but less than or equal to two below you get a D. More than two below you get an F.

For the mean we use a simple arithmetic mean (average). The standard deviation, also called the root mean square (RMS) is square root of the average of the squares of the deviations from the mean. Written as a formula it is the following:

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} (x_i - m)^2}{N}}$$

Here m is the mean, N is the number of grades in the class, and x_i is the ith grade in the class. Your task for this problem is to read in all the grades for a class and tell us what letter grade each number grade gets.

Input: The input for this problem consists of a header line that tells you how many different classes you will be doing this for. Each class will have a line telling you how many students are in the class followed by a single line giving you the grades. The grades will be separated by a single space.

Output: For each class you will output a header line saying "Class #" where # is the number for that class starting at 1. You will then give each number grade and the letter grade it gets separated by a space. The grades should be in the order they appeared on the input line.

Sample Input:
```
2
4
95 85 75 65
5
40 60 50 30 20
```

Sample Output:
```
Class 1
95 A
85 B
75 C
65 D
Class 2
40 B
60 A
50 B
30 C
20 D
```

Problem 2 – Will It Fit?

Your sister is planning a two week trip to Europe. Wanting to be prepared for whatever might arise, she has placed virtually the entire contents of her closet on her bed. Unfortunately for her, she can't take every suitcase in the house on the trip. Your task is to figure out whether a certain set of her belongings can fit into a set of suitcases.

To keep things simple everything is a scalar. You don't have to worry about how stuff is arranged. Each suitcase has a volume and each item takes up a certain amount of volume. All the volumes will be integer values. Note that this isn't as simple as adding the volumes of the suitcases and the objects to pack. Each object has to fit completely in one case, you can't split an object between two cases. You also can't overfill the suitcases. So if you had 5 suitcases of size 3 and 6 objects of size 2 you can't make it fit. You can hold 15 units of volume, but once you have put one object of size 2 in each suitcase the 6th object can't fit in any of them.

Input: The input for this problem consists of a header line that tells you how many configurations of suitcases and objects you will have to test. Each configuration will have 3 lines. The first line will have the number of suitcases followed by the number of items to pack. The second line will be the volumes of the suitcases. The third line will be the volumes of the items to pack.

Output: For each configuration you will either print "It fits! Off you go!" or "Bummer, won't go."

Sample Input:
```
2
5 6
3 3 3 3 3
2 2 2 2 2 2
2 3
5 7
4 3 5
```

Sample Output:
```
Bummer, won't go.
It fits! Off you go!
```

Problem 3 – Toll Road Turmoil

San Antonio is in the throws of battle over whether to add toll roads. You figure it is inevitable that they will appear, and when they do you want to be on top of the market for assisting people in their speedy passage. You need to write a program that keeps track of how much change a person has in his/her car and will tell them if they can pay a certain toll and if so, what combination of coins they should use so that they use the minimum number of coins. They must have exact change. So if the toll is 25 and they only have 3 dimes they have to take the slow way around. Note that you have pool of change for all tolls and each toll paid takes coins away.

Input: The input begins with a single line telling you how many quarters, dimes, nickels, and pennies the driver starts with. After that is a line with the number of toll roads that driver will come to. Each toll road has a single line that is the amount of money needed to get through that toll road, given as a number of cents.

Output: For each of the toll roads you will either output “Looks like the slow road for you.” if they don't have enough money, or tell them what combination of quarters, dimes, nickels, and pennies they should use. These numbers should be on a single line with quarters followed by dimes, nickels, and pennies with a single space between each number.

Sample Input:

```
4 3 4 6
5
1025
78
27
35
20
```

Sample Output:

```
Looks like the slow road for you.
3 0 0 3
1 0 0 2
0 3 1 0
Looks like the slow road for you.
```

Problem 4 – Hitting the Trail Mix

You and a bunch of friends are planning an extended hiking trip. You have been put in charge of getting the trail mix. Since this is an extended trip you have gotten your hands on bags that can hold 10 lbs of dry mix. You are heading to the bulk food section of your favorite store to make the purchase. This particular store is your favorite because they are cheap and in this case that means every bulk item costs the same amount of money. (You're young so potential issues of quality don't really bother you.) Given this, your real deciding factor is how much you and your friends like different things. You find that out from them and fill their bags in order. This means the people you fill last might be out in the cold, but if they really cared you figure they should have bought their own mix.

Input: The input for this problem will start with a single line that has two numbers, one telling you how many people are going on the trip and a second telling you how many bulk items your store has. For each item there is one line giving the name of the item and how many lbs of it the store has. After that, each person has one line giving his/her name followed by how many items they specified a preference for. There is one line for each item giving the item name and an integer score for how much they like it. The score will have no ties.

Output: You will process each person in turn and print out one line for each of them. That line will give how much of each item was put into their bag. Use the format shown in the sample output below. They should never take more than 10 lbs. If the store runs out of everything they want before their bag is full they might have fewer than 10 lbs.

Sample Input:

```
2 4
peanuts 5
M&Ms 4
raisins 8
pretzels 13
Glenn 3
raisins 5
peanuts 10
M&Ms 8
Mark 3
M&Ms 10
pretzels 2
raisins 6
```

Sample Output:

```
Glenn – peanuts (5), M&Ms (4), raisins (1)
Mark – raisins (7), pretzels (3)
```

Problem 5 – Stacked Transfer

Your friend Bill got a summer job working in the library. The alternative was a job with a moving company and Bill didn't see couch carrying as being his strong suit. Unfortunately for Bill, had he found out why they wanted him he might have gone with the moving job. Turns out the library is renovating over the summer which means that the books need to be moved out of each area they are renovating only to be moved back later. You are going to write a program to help figure out how many trips Bill will be taking.

The way the process will work is that one of the other library employees will take books down from the shelf and put them in stacks. Because librarians are rather particular about the order of the books on their shelves, Bill has to carry the books in the same stacked order he gets them in. So he can pick up any number of books from a stack and carry it, but he can't mix books around to make things more efficient. Remember, Bill wasn't into couch carrying and the real limit is that he can only carry 20 lbs at any one time. Any more than that and he will throw out his back. So if the top three books in a stack weigh 10 lbs, 5 lbs, and 7 lbs, Bill will take the top two in one trip then come back to get the 3rd and possibly others below it.

Input: You are first given a number for the number of stacks of books Bill has to move. There will be one line of input for each stack. That line has 1 or more numbers on it giving the weights of the books in the order they are stacked. The numbers are integers and are separated by one space each.

Output: For each stack of books you will simply output how many trips Bill has to take to carry it. If a stack contains any books that weigh more than 20 lbs you output, "Call in the fork lift!"

Sample Input:

```
3
10 10 10 10 10 10
7 5 6 3 7 4 8 2 6 5 3 6 2 7 1 6 3
3 5 2 40 16 7
```

Sample Output:

```
3
5
Call in the fork lift!
```

Problem 6 – A Monkey Could Do That

Inevitably, you have heard the saying that given enough time and enough monkeys with enough typewriters you could eventually produce the full works of Shakespeare. For this problem you will figure out just how long that would take. To keep things simple we'll assume that we only have one monkey and that monkey randomly hits a key or the space bar ten times each second. We'll also do something a bit smaller than the full works of Shakespeare. Instead, you will be given one line and asked what the expectation time is for how long it would take the monkey to enter that string. All keys are equally likely except for the space bar because it is bigger. Assume we have a keyboard with no keys other than letters and the space bar, so each letter has an equal probability of 1 in 30 and the space bar has a probability of 4 in 30.

As a simple example, assume we just want to know how long it would take for the monkey to type in “a”. Well, each key stroke there is a 1 in 30 chance that it will be an “a”. So after 30 keystrokes we would expect that the money would have hit an “a” and because he hits 10 keys each second we expect to wait 3 seconds. Obviously it might take much longer, but we just want an expectation time. Were the string to be “is” then we would expect a 1 in 30 chance of the “i” and 1 out of 30 of those should be followed by the “s”. So our expected waiting time is 90 seconds.

Input: The input begins with the number of lines you have to consider. After that will be the proper number of lines, each with a string consisting of letters and spaces.

Output: For each line you will output how many seconds we expect to wait before that line would be typed. The number does not have to be an integer.

Sample Input:

```
3
is
this
going to work
```

Sample Output:

```
90.0
81000.0
9.96451875e16
```

Problem 7 – (I'll See That)++

Little Timmy is only ten years old, but he loves watching his father play poker every Tuesday night with his friends from work. Sometimes, Timmy's dad will let him sit in his lap as they play, and Timmy has begun to learn what certain hands are called. He knows that if his dad has all hearts or diamonds, it's called a flush, and if the cards are all in a direct sequence of numbers it's called a straight. Timmy can even recognize a royal flush, as his dad always tells him it's the one hand he's never gotten. What challenges Timmy is seeing the other, less obvious combinations. Write an algorithm to tell Timmy whether a given hand has a full house, four of a kind, three of a kind, two pair, one pair, or is just a high card.

Input: The input will begin with a single number, n, on a line telling you how many hands you will examine. The next n lines will have 5 cards listed on them. A card is represented by a letter for the suit followed by a number. The number will be between 1 and 13.

Output: The output will consist of n lines where you tell Timmy what is in the hand. The options you will print from are as follows: "High Card", "Pair", "Two Pair", "Three of a Kind", "Full House", "Four of a Kind".

Sample Input:
```
3
C2 H3 D3 H12 S3
H1 S11 D4 S9 S10
D5 C13 S13 H5 H13
```

Sample Output:
```
Three of a Kind
High Card
Full House
```

Problem 8 – You Can't Get There From Here

You have stopped in a small town in west Texas to ask for directions. There is an old man sitting at the dusty gas station giving you some directions. Unfortunately, his description is less than ideal. He is able to list the local towns and which highways connect one town to another, but it is up to you to figure out which highways you should take to get from where you are to where you want to go. Even more unfortunate, the listing you get isn't quite complete. For this problem, you just have to be able to tell whether you can get from you current location to other towns based on the list of highways the old man has listed.

Input: The input begins with a line that has the name of the town you are currently in followed by the number of highways that have been listed to you. The lines after that give you a highway name and the names of two towns separated by spaces. Following the list of highways is a line with the number of towns you will be asked to check. Each line after that has a name of a town. The listed towns might not be connected to any of the listed highways. All towns and highways have names that are single words with no spaces.

Output: For each of the towns at the end of the file you will print out one line. If you can get from your current town to that town you print "Just go a ways over there." If there is no path connecting them, print out "You can't get there from here."

Sample Input:

```
Dustville 6
TX304 Dustville Texline
TX285 Texline Tumbleweed
US14 Tumbleweed Bend
US87 Texline Armadillo
I-10 Stockton ElPaso
CR875 Bend Texline
5
Texline
Armadillo
Dinky
Stockton
Bend
```

Sample Output:

```
Just go a ways over there.
Just go a ways over there.
You can't get there from here.
You can't get there from here.
Just go a ways over there.
```

Problem 9 – Save the Lobsters!

The state of Maryland has noticed significant decreases in their local lobster population. To combat this the state has passed a new law limiting the number of lobsters each ship can bring in for the days catch. Basically they decided that no ship can bring in more than 1000 lobsters each day. You are supposed to read catch records for ships and determine if they are in violation of the law or not.

Input: The input will begin with a line that has a single number telling you how many ships you will be checking. Each line after that contains the number of lobsters caught by a single ship.

Output: For each ship you should print out either “legal” or “violation” with one per line.

Sample Input:
```
4
24
1584
998
19583
```

Sample Output:
```
legal
violation
legal
violation
```