PSCSTA December 2011 Contest

Novice Problems

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

Number	Name	Started	Solved
Problem 1	Census		
Problem 2	Greetings		
Problem 3	Grocery		
Problem 4	Triangles		
Problem 5	Closest		
Problem 6	Grandparents		
Problem 7	Swap		
Problem 8	Exit		
Problem 9	Distance		
Problem 10	Scanner		
Problem 11	Summation		
Problem 12	Wimbledon		

Good luck!

Problem 1. - Census

Input File: census.dat

General Statement: In its recent 2010 process, the U.S. Census Bureau claimed that the population of the United States was 308,745,538. In order to expedite the census process, the bureau had to decide how many people to hire to do the counting. Each census clerk could only count a certain number of residents each day. The Census Bureau had to calculate how many days it would take a single clerk to complete a census of their assigned area, given the average count per day for that clerk, and the estimated number of residents in that area.

Assume that on each day, the clerk counted the same number of residents, except on the last day when it was possible to finish early since there could be fewer residents remaining than the daily average.

Input: In the data file, the first number N represents N data sets to follow. Each data set consists of two integers, **n** and **p**, where **n** is the number of people that a single clerk can count in a day and **p** is the estimated number of residents.

Constraints: **`n >= 1(one), 1 <= p <= maximum integer value`**

Output: Your program must output the number of days it will take the clerk to completely count the estimated number of residents.

Sample Input :
```
5
100 9900
27 1379
99 103
812 406
3 471834
```

Sample Output:
```
99
52
2
1
157278
```

Problem 2. - Greetings

Input File: none

General Statement: Write a program to prompt the user for his or her name and greet him or her.

Input: The user will type their first name followed by the "Enter" key when prompted. The user will always input only a single word. Note that there is a space after the colon in the prompt text.

Output: A greeting in the format "Hello, **name**, welcome to the competition!"

Sample Input (user input in italics and underlined, program-generated text not):
```
Your name: Jacquie
```

Sample Output:
```
Hello, Jacquie, welcome to the competition!
```

Sample Input (user input in italics and underlined, program-generated text not):
```
Your name: Julius
```

Sample Output:
```
Hello, Julius, welcome to the competition!
```

Problem 3. - Grocery

Input File: grocery.dat

General Statement: Your best friend gives you a grocery list and you start your shopping at S-Mart, the store that only sells foods starting with the letter S. After you finish shopping at S-Mart, you decide to print your grocery list and cross out all items starting with S by surrounding them with dashes.

Input: In the data file, the first number N represents N grocery items to follow. Each line contains one grocery item. Every item is one word long and starts with a capital letter.

Output: Print the items from the input file in the same order and format as the input file, but include a dash "-" before and after each item starting with S.

Sample Input:
```
7
Milk
Spaghetti
Apple
Hamburger
Cookies
Spinach
Soda
```

Sample Output
```
Milk
-Spaghetti-
Apple
Hamburger
Cookies
-Spinach-
-Soda-
```

Problem 4. - Triangles

Input File: triangles.dat

General Statement: Your friend gives you a file that has a series of characters and numbers that she wants printed in triangles. Each character is followed by a positive number that corresponds to the longest lines in the triangles. Read in each character and number and print two triangles: one which ascends from a single character up to a height of X characters, and the next immediately after, starting at a height of X characters and descending down to 1 character.

Input: In the data file, the first number N represents N data sets to follow. Each data set contains a character C followed by a space followed by a number X.

Output: For each data set in the input file, print a triangle of height X containing the character C. Each triangle starts and ends with one character C. The two longest lines in each triangle are in the middle and contain the character C X times.

Sample Input:
```
3
* 3
A 5
Z 2
```

Sample Output:
```
*
**
***
***
**
*
A
AA
AAA
AAAA
AAAAA
AAAAA
AAAA
AAA
AA
A
Z
ZZ
ZZ
Z
```

Problem 5. - Closest

Input File: closest.dat

General Statement: You have been hired to help create the next generation of mobile local business finders. This advanced system will find all establishments of a specified type within a 100-mile radius of the user and suggest the closest one. The user will also be able to specify the kind of establishment he or she is looking for.

Input: The first line of each data file specifies the number N of businesses in the file. The next line describes the category of the businesses provided as a single word (restaurants, gyms, etc). N lines, each representing a business, follow. Each line starts with a distance in miles from the user's location followed by a single space, followed by the business name in one word. The file will only contain one set of businesses of the same category.

Output: The word "Closest" followed by a space, followed by the business category, followed by a colon and a space, followed by the name of the business closest to the user's current location. In the case of a tie, the business appearing first in the file should be displayed.

Sample Input:
```
4
Groceries
1 Safeway
.5 QFC
1 WholeFoods
.75 TraderJoe's
```

Sample Output:
```
Closest Groceries: QFC
```

Problem 5. - Grandparents

Input File: grandparents.dat

General Statement: You live in Seattle, Washington, and you have two sets of grandparents: one set of grandparents lives in Chicago, and the other lives in New York. On every holiday, your mom and dad make you call your grandparents and wish them a happy holiday. However, the Chicago grandparents have a different schedule than the New York grandparents, and you and both sets of grandparents live in different time zones, so it is hard to tell if they are free to call! You must write a program that will print whether or not both sets of grandparents are available. Here is some necessary information:

Grandparents A:
- Live in Chicago
- Go to bed at 20:00
- Wake up at 8:00
- Leave the house on weekdays between 15:00 and 16:00 for a pottery class.

Grandparents B:
- Live in New York
- Go to bed at 22:00
- Wake up at 9:00
- They go salsa-dancing weekends between 19:00 and 20:00

NOTE: All times are given in military time (24-hour clock) – 22:00 is 10pm and 10:00 is 10am.

The input file will tell you what time it is in Seattle in military time, and whether or not it is a weekday. You will need to be able to convert between Seattle time, Chicago time, and New York time. Here is some helpful information:

	Location (City)		
	Seattle	**Chicago**	**New York**
Time	**12:00**	**14:00**	**15:00**

Input: The first number in the input file will represent how many data sets will follow. Each data set will be on its own line and it will have three tokens. The first token will be a string (either "WEEKEND" or "WEEKDAY") that indicates whether it is a weekday or a weekend. The next two tokens represent the current time IN SEATTLE. The first of the two tokens will be an integer representing the hour, and the second token will be an integer representing the minutes.

Output: Your program should output one of two things. If both of your grandparents are available, then your program should print: "Yay! I can call my grandparents!" If either one or both sets of grandparents are unavailable, your program should print, "Dang! I have to wait to call my grandparents."

Sample Input:

```
5
WEEKEND 16 30
WEEKDAY 1 21
WEEKDAY 15 47
WEEKEND 20 05
WEEKEND 7 00
```

Sample Output:

```
Dang! I have to wait to call my grandparents.
Dang! I have to wait to call my grandparents.
Yay! I can call my grandparents!
Dang! I have to wait to call my grandparents.
Yay! I can call my grandparents!
```

Problem 6. - Swap

Input File: swap.dat

General Statement: You have been asked to write the next great American pop song. Since you are not feeling lyrically inspired, you decide to write a program to create new songs out of existing ones. You will swap every two lines of a given song.

Input: The data file's first number represents the number of lines of input to follow. The rest of the file will represent lines from a pop song.

Output: Your program should switch the order of the lines in the file in a pairwise fashion. The first two lines should be swapped, then the next two and so on. If there are an odd number of lines in the file, the final line is not moved.

Sample Input:
```
4
And I was like baby, baby, baby, oh
Like baby, baby, baby, no
Like baby, baby, baby, oh
I thought you'd always be mine, mine
```

Sample Output:
```
Like baby, baby, baby, no
And I was like baby, baby, baby, oh
I thought you'd always be mine, mine
Like baby, baby, baby, oh
```

Problem 7. - Exit

Input File: none

General Statement: Airplanes have special doors to be used in case of an emergency during a flight. GetThere Airline has noticed that passengers get confused about which emergency exit they should use. The company has decided to suggest the best emergency exit to take on each passenger's ticket. You are tasked with writing the code to determine a passenger's ideal exit based on their assigned seat.

Passengers in rows 1 through 14 (inclusive) should use a front exit. Passengers in rows 15 through 28 should use a middle exit. Finally, passengers in rows 29 through 42 should use a back exit. Passengers in seats A, B or C should use an exit on their left. Passengers in seats D, E or F should use an exit on their right.

Input: The user will type first their row number then their seat number when prompted. Note that there is a space after each question mark in the prompt text. The user will always provide a row number between 1 and 42 (inclusive) and will always provide a capitalized seat letter between A and F (never lowercase and never more than a single letter).

Output: The program will start by prompting for the user's row as well as his or her seat, as shown below. Once these are entered, the program will output best emergency exit to use in the following format: "Please use the **location direction** exit." **Location** is one of front, middle or back. **Direction** is left or right.

Sample Input (user input in italics and underlined, program-generated text not):
```
What row are you in? 12
What seat are you in? B
```

Sample Output:
```
Please use the front left exit.
```

Sample Input (user input in italics and underlined, program-generated text not):
```
What row are you in? 12
What seat are you in? E
```

Sample Output:
```
Please use the front right exit.
```

Sample Input (user input in italics and underlined, program-generated text not):
```
What row are you in? 29
What seat are you in? C
```

Sample Output:
```
Please use the back left exit.
```

Problem 8. - Distance

Input File: none

General Statement: You are creating a program to help users know how long it will take them to reach a destination.

Input: At the first prompt, the user will input a distance in miles. At the second prompt, the user will input a speed in miles per hour. Note that there is a space after each question mark in the prompt text.

Output: The program will start by prompting for the user's distance from their destination as well as their speed, as shown below. Once these are entered, the program will output the time until the destination is reached using the following format: "You are **minutes** minutes from your destination." The number of minutes is rounded to the nearest whole number of minutes.

Sample Input (user input in italics and underlined, program-generated text not):
```
How far are you from your destination (in miles)? 5
How fast are you going (in miles per hour)? 10
```

Sample Output:
```
You are 30 minute(s) from your destination.
```

Sample Input (user input in italics and underlined, program-generated text not):
```
How far are you from your destination (in miles)? .5
How fast are you going (in miles per hour)? 60
```

Sample Output:
```
You are 1 minute(s) from your destination.
```

Sample Input (user input in italics and underlined, program-generated text not):
```
How far are you from your destination (in miles)? 30
How fast are you going (in miles per hour)? 65.6
```

Sample Output:
```
You are 27 minute(s) from your destination.
```

Problem 9. - Scanner

Input File: scanner.dat

General Statement: Recently airports have begun to install full-body scanners at security checkpoints. You have been hired to create a program that will read a passenger's boarding pass and convert their name to be read by the scanner. To ensure privacy, only their last name and first initial will be recorded.

Input: In the data file, the first number N represents N data sets to follow. Each data set consists of the name of a passenger: first name, then a space, then the last name.

Output: Your program must output the last name and first initial of each passenger in the following format : last name, first initial.

Sample Input :
```
5
JOHN DOE
MARY JONES
ROBERT MOORE
DOUG WILSON
MARIA WILLIAMS
```

Sample Output:
```
DOE, J.
JONES, M.
MOORE, R.
WILSON, D.
WILLIAMS, M.
```

Problem 10. - Summation

Input File: summation.dat

General Statement: You have been hired by the European Organization for Nuclear Research (CERN) to do some base-level coding on their new Hadron Collider. Your job is to generate some test data using the following summation formula.

Using two integers **k** and **n,** your program must calculate and output the result of the following summation equation:

$$\sum_{k}^{n} k \text{ where } \frac{x(x+1)}{2}$$

Where k=1 and n=5, the generated value would be 35, as shown below.

$$\begin{matrix} k = 1 \\ n = 5 \end{matrix} \rightarrow \frac{1(2)}{2} = \mathbf{1}, \quad \frac{2(3)}{2} = \mathbf{3}, \quad \frac{3(4)}{2} = \mathbf{6}, \quad \frac{4(5)}{2} = \mathbf{10}, \quad \frac{5(6)}{2} = \mathbf{15}$$

$$1 + 3 + 6 + 10 + 15 = 35$$

Input: In the data file, the first number N represents N data sets to follow. Each data set is a pair of integers, **k** and **n**.

Output: The test value generated by each pair of integers.

Sample Input :
```
4
1 5
2 7
3 4
9 12
```

Sample Output:
```
35
83
16
244
```

Problem 11. - Wimbledon

Input File: wimbledon.dat

General Statement: The longest tennis match in history took place in the first round of the 2010 Wimbledon tournament. American John Isner needed 183 games to defeat Nicolas Mahut of France (the final set score was 70-68), in a match that lasted 11 hours and 5 minutes, spanning three days. There were 980 points overall, and Mahut won more, 502-478. There were 711 points in the fifth set, and Mahut won more, 365-346. But Isner won the most important point of all: the last one, which happened to be a rather nondescript backhand winner down the line. The Wimbledon officials have decided to computerize their verbal scoring system and need your help to correctly express the set score. Given a list of set scores, write a program to output the correct verbal set score.

An interesting curiosity in tennis is the expression for a score of zero, which is verbalized as **"love"**, derived from the French noun "l'oeuf", which literally means "goose egg". This word is pronounced "luff", and over the years "morphed" into the English word "love".

Input: In the data file, the first number N represents N data sets to follow. Each data set contains two values, separated by a dash. The word "set" is verbalized when a player has won 6 or more games, by a margin of 2 or more, otherwise the actual score is called out. **Note:** Despite the record setting 70-68 set score chronicled above, it is guaranteed for the purposes of this exercise that no score will exceed 9 games for either player.

Output: The score of games within a set is verbalized in the ordinary manner, such as **"three - six"** for a score of **3-6**, and "set" if the set is complete. Ties, such as 3–3, are verbalized as "three all". Scores of zero are to be called out as **"love".**

Sample Input :
```
7
3-3
7-5
0-4
1-0
3-6
5-2
7-8
```

Sample Output:
```
three all
set
love - four
one - love
set
five - two
seven - eight
```