A+ Computer Science
November / December 2011

Computer Science Competition
Hands-On Programming Set

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.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

Number	Name	Started	Solved
Problem 1	Bed Bugs		
Problem 2	Census		
Problem 3	Euros		
Problem 4	Face		
Problem 5	HourGlass		
Problem 6	Neutrinos		
Problem 7	In The Red		
Problem 8	Full Body Scanner		
Problem 9	Spill		
Problem 10	Summation		
Problem 11	Tiger		
Problem 12	Wimbledon		

Good luck!

Problem 1 - Bed Bugs

Program Name: bedbugs.java **Input File: bedbugs.dat**

General Statement: Due to the recent bed bug outbreak the National Pest Management Association (NPMA) has decided to track their breeding. Their research says that bed bugs reproduce at an astonishing rate. Your job is to simulate their breeding habits by producing an output grid of their progress over several cycles of breeding.

As bed bugs breed, they spread in four directions (North, South, East, and West), but for the purposes of this simulation, only a single bed bug can occupy a grid section at any one time. If two bed bugs try to occupy a single section, one bed bug defeats the other, battling to the death. If a bed bug spawns outside of the grid, it falls off the bed and dies. Once a bed bug reproduces, it dies as well. Of course, it is entirely possible that the intial bed bug misses the bed entirely, fallling to its death, leaving the bed totally free of the nasty creatures.

Input: The input file contains an integer N, followed by N sets of data. Each data set contains a string indicating the size of the bed ("twin","full", "queen", or "king"), two integers x and y indicating the location of the initial bed bug, and n, the number of breeding cycles you are to simulate. For our purposes, a twin bed measures 3X6 in feet, a full size bed 4X6, queen 5X7, and king 7X7. Also, each square foot contains at most one bed bug.

Output: Your program must output the **nth** generation of population (The input is considered the first generation). Remember, only a single bed bug can occupy any one space at a time, and bed bugs that fall outside of the grid, fall off the bed and die. Provide at least one blank line after the output of each "bed".

Sample Input:

```
3
twin 4 1 1
king 2 5 3
full 6 4 2
```

Sample Output:

```
...
...
x..
.x.
x..
...

..x.x.x
.x.x.x.
..x.x.x
...x.x.
....x..
.......
.......

....
....
....
...x
..x.
.x.x
```

Problem 2 - Census

Program Name: census.java **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. Since each census clerk could only count a certain number of residents each day, the Census Bureau had to calculate out 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 3 - Euros

Program Name: euros.java **Input File: euros.dat**

General Statement: Fantasy Football has spread overseas. This year, the Chancellor of Germany finds himself in the final bracket of his league and is hoping to win some prize money. The only problem is that the prize is in US dollars, and he needs it in Euros. Your job is to convert the prize amounts from USD to Euros. Use the exchange rate of:

1 USD = 0.7413 Euros

Input: The input file will contain the prize amounts in USD for the first six places in the league.

Output: Your program must output the amount of the winnings rounded to the nearest $1/100^{th}$ Euro, and in European currency format.

Format requirements:

- Include the prefix EUR
- Round and show the fractional portion of the value to 2 decimal places
- Output in the customary European format of currency display, with periods in place of commas, and vice versa.

An error tolerance of +- .01 will be allowed.

Sample Input :

```
4
$2761543.97
$184313.18
$94315.64
$348.19
```

Sample Output:

```
EUR 2.047.132,54
EUR 136.631,36
EUR 69.916,18
EUR 258,11
```

Problem 4 - Face

Program Name: face.java **Input File: none**

General Statement: Create a picture, using only keyboard characters, of your face or the face of one of your teammates, so that the judges can recognize you when you accept your award today. The only restrictions are:

- it must between 5 by 5 and 10 by 10 characters in total size.
- It must contain at least one \ and one “
- It cannot be the face shown in the example…that one belongs to one of the judges!

Input: None

Output: A character picture of your face!

Sample Output:

```
\\\\////
( ○   ○ )
 \  ^   /
   --
   ""
```

Problem 5 - HourGlass

Program Name: hourglass.java **Input File: hourglass.dat**

General Statement: Create and oputput a star pattern of an hourglass that has just been flipped over and is most of the way through the process, with some of the sand still in the top section, and the rest on the bottom. The number of "layers" of sand is directly proportional to the size of the glass, and always the same for the top and bottom sections. In the examples below, the size 7 and 9 hour glasses each have 1 layer of sand in the upper section, and one layer in the lower. The size 11 glass has two layers each, and the size 19 glass three layers.

Input: N odd integers indicating the size of the hourglass to be created, each integer ***i*** in the range **7<= *i* <= *21***.

Output: For each integer size create an hourglass pattern, as described above and shown in the examples. One blank line should follow each output pattern.

Sample Input:

```
7 19 9
```

Sample Output:

```
*******
 *   *
  ***
   *
  * *
 *****
*******

*******************
 *                 *
  *               *
   *             *
    *           *
     *         *
      *       *
       *******
        *****
         ***
          *
         * *
        *   *
       *     *
      *       *
     *         *
    *           *
   *************
  ***************
 *****************
*******************

*********
 *       *
  *     *
   ***
    *
   * *
  *   *
 *******
*********
```

Problem 6 - Neutrinos

Program Name: neutrinos.java **Input File: neutrinos.dat**

General Statement: A group of physicists known as OPERA, short for Oscillation Project with Emulsion-Tracking Apparatus, working near Geneva, Switzerland, reported in a recently published paper that they had measured neutrinos traveling at speeds in excess of Einstein's commonly accepted cosmic speed limit, the speed of light. According to scientists familiar with the paper, the neutrinos raced from a particle accelerator outside Geneva, where they were created, to a cavern underneath Gran Sasso in Italy, a distance of about 450 miles, about 60 nanoseconds faster than it would take a light beam. That amounts to a speed greater than light by about 0.0025 % (2.5 parts in a hundred thousand). A **neutrino** is an electrically neutral subatomic particle, with a small but non-zero mass, similar to an electron, generated by the sun. Being electrically neutral, thus unaffected by the electromagnetic spectrum to which light beams belong, it is able to pass through ordinary matter virtually unaffected, "like a bullet passing through a bank of fog". The neutrino (meaning "small neutral one") is denoted by the Greek letter **ν** (nu).

To help these scientists verify their results with more data, calculate the time that light would take to travel various distances in miles, and the time that these neutrinos would take, at the same relative rate of 0.0025 % faster (according to the claims of the OPERA group), and output the positive difference in nanoseconds.

For the purposes of this exercise, the definition of the speed of light is 299,792,458 meters per second, and one second equals 10^9 nanoseconds. One meter is defined as 39.37 inches, and there are 5280 feet in a mile.

Input: In the data file, the first number N represents N data sets to follow. Each data set is a distance in miles.

Output: For each mileage distance, output the calculated time difference in nanoseconds as described above. An error tolerance of +-.01 will be allowed.

Sample Input :

```
5
450
1000
10000
34
93000000
```

Sample Output:

```
60.39
134.21
1342.05
4.56
12481075.44
```

Problem 7 - In The Red

Program Name: red.java **Input File: red.dat**

General Statement: In accounting, balance sheets are kept to track how much money a person has in an account. Your job is to create a simple three column balance sheet, showing credits in one column, debits in the next, and a running balance in the third. If the balance dips into the negative, you must show the value in parentheses, the standard method in accounting.

Input: The first value will be the opening balance in the account, followed by an integer N indicating N sets of data to follow. Each subsequent data set will be either the letter 'C' for credit, or 'D' for debit, followed by a dollar amount.

Output: Output a complete balance sheet as shown below, with headings and boundary lines above and below as shown, and columns 14 spaces wide. All amounts will be at least one dollar, and are guaranteed to "fit" in the balance sheet. All values with four digits or more in the whole number portion are to be output with "comma" notation, and all negative values in the balance column are to be shown in parentheses.

Sample Input:

```
999.99
5
C 8250.55
D 12397.45
D 12345.67
C 15492.59
C 5000000
```

Sample Output:

```
         DEBIT         CREDIT        BALANCE
-------------*-------------*-------------*
                                $      999.99
               $    8,250.55 $    9,250.54
$   12,397.45                 $  (3,146.91)
$   12,345.67                 $ (15,492.58)
               $   15,492.59 $        0.01
               $5,000,000.00 $5,000,000.01
-------------*-------------*-------------*
```

Problem 8 - Full Body Scanner

Program Name: scanner.java **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 last name - with one space of separation.

Output: Your program must output the last name and first initial of each passenger in the format below – last name, comma, first initial, period:

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.
```

A+ Computer Science – November Problem Set

Problem 9 - Spill

Program Name: spill.java **Input File: spill.dat**

General Statement: Oil company giant JP has another spill on their hands. You are a part of a new company in charge of cleaning up the spill with a state-of-the-art cleaning robot named Hayward. To clean up a spill, Hayward is dropped from a helicopter, hopefully right into the very center of a square spill grid (northwest corner is position 1,1). It is your job to simulate Hayward's movements as he cleans up the spill. Curiously, he can only move forward or turn left, just like another robot you might be familiar with. Hayward's built-in logic is to turn left when there is oil to his left (but not otherwise), creating an efficient spiral path in his efforts to clean up all of the spill. He will stop cleaning when he detects no more oil to clean inside his designated grid, and he will not revisit a location he has already cleaned. Also, the helicopter flyer is enough of an ace that he never misses dropping Hayward into the grid, but doesn't always get him into the very center where he is most efficient, which results quite often in a partially cleaned grid.

Input: The input file will contain an initial value N, with N lines of data, each in the format: **n x y**, where **n** (**n >= 3**) is the number of columns and rows of the square-shaped oil spill grid, and **x y** is the row and column where Hayward will be dropped, initially facing North.

Output: Output a map of the path that Hayward will travel using **"^", "<", "v" ,">"** to show which direction Hayward just moved. The first move will always be North **("^")**. The character **'*'** will dictate an uncleaned area, one that Hayward was not able to reach, the capital letter "O" will indicate the spot where the robot was initially dropped, and the capital letter "X" marks the spot where he stopped cleaning.

Sample Input:

```
3
7 4 4
4 3 2
5 5 5
```

Sample Output:

```
<<<<<^X
v<<<^^^
vv<^^^^
vvvO^^^
vvv>>^^
vv>>>>^
v>>>>>>

X<^*
<^^*
vO^*
v>>*

*****
*****
*****
***<^
***XO
```

Problem 10 - Summation

Program Name: summation.java **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.

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

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

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 - Tiger

Program Name: tiger.java **Input File: tiger.dat**

General Statement: Due to the recent turmoil in his personal life, Tiger Woods has hired your public relations firm to help him discover whom he can trust; he is extremely paranoid and at this point trusts only his closest friend, who is first on the list of other close friends for you to interview. He very much wants to trust the others, but at this point thinks they are all liars. Your job is to visit with these few, each of whom will tell you who they think is lying, and to discern who indeed is trustworthy based on these interviews. Each person you interview will give you a list of one or more people they think are liars.

Assume that the people you interview are all liars, except for the first one, Tiger's closest friend, whom he trusts to tell the truth. For example, in the input shown below, Stephen (the closest companion who is undoubtedly trustworthy and who is interviewed first) declares Thomas to be a liar, and Thomas' false claim that Gilbert lies implies that Gilbert is indeed trustworthy. George, who accuses Stephen of lying, proves himself a liar with that claim. Furthermore, he accuses Ethan of being a liar, which infers Ethan to be a truthteller. Both Ethan and Stephen declare Thomas a liar, which verifies Stephen's earlier assertion and validates each of them as trustworthy. Therefore, the trustworthy companions in this case are Ethan, Gilbert and Steven.

Input: In the data file, the first number N represents N companions in Tiger's circle of friends, which are listed in interview order at the beginning of the data file. Each person interviewed is followed by an integer **n**, which is then followed by **n** lines, each containing the name of another person in the group that the accuser says is a liar.

Output: In alphabetical order, output the list of trustworthy companions.

Sample Input:

```
5
Stephen Thomas George Ethan Gilbert
Stephen 1
Thomas
Thomas 1
Gilbert
George 2
Stephen
Ethan
Ethan 1
Thomas
Gilbert 1
Thomas
```

Sample Output:

```
Ethan
Gilbert
Stephen
```

Problem 12 - Wimbledon

Program Name: wimbledon.java **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
```