A+ Computer Science
November-December 2012
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.
6. 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
Problem 1	Baseball Bin Berry Berry Good To Me
Problem 2	Box J
Problem 3	Crazy Conversions
Problem 4	Day To Day
Problem 5	Flex J Box
Problem 6	Gender Race
Problem 7	Maze Map
Problem 8	Snapshot Day
Problem 9	Taxi
Problem 10	Test Parity
Problem 11	Weird Change
Problem 12	Y Tree

Good luck!

1. Baseball Bin Berry Berry Good To Me

Program Name: baseball.java Input File: baseball.dat

General Statement : For you diehard SNL fans, you will recognize this (slightly politically incorrect) quote from Chico Escuela, the fictional baseball player character introduced by Garrett Morris in the fourth season back in 1978, during a hilarious interview with Jane Curtin (or "Hane", as he pronounced her name) in the Weekend Update sketch for that episode.

Nevertheless, Chico's manager needs your help with the scorecard he keeps after every play. You will be given a status of what runners are currently on base, and then the type of hit the batter makes (0 = out, 1 = single, 2 = double, 3 = triple). The current status of the runners on base will be indicated by a 1 or 0, 1 meaning on base, 0 showing the base is empty. An input line of 1 0 1 2 means runners are on 1st and 3rd, and the batter hits a double, resulting in an output line of 0 1 1 1, meaning the batter reached 2nd, runner is still on 3rd, and 1 run scored. First base is empty.

Input: Several sets of four integers as described above, with one space between each, all on one line.

Output: The resulting outputs for the given sets, also as described above, with at least one space between each value, all on one line.

Example Input File

```
1 1 0 0
1 1 1 2
0 0 1 1
1 1 1 4
```

Example Output to screen:

```
1 1 0 0
0 1 1 2
1 0 0 1
0 0 0 4
```

2. Box J

Program Name: BoxJ.java Input File: none

General Statement : Output this 4X5 box of stars with the letter J imbedded as shown.

Input: None

Output to screen:

```
*****
*****
**J**
*****
```

3. Crazy Conversions

Program Name: conversions.java Input File: conversions.dat

General Statement : Flash, the mad mathematician, needs your help with some tricky formulas. In a momentary loss of computing ability, he needs you to compute the following strange formulas into their final value. In his madness, he chose to name his formulas after his pets, Crash, Dash, Mash, and Trash.

Crash = $\frac{A}{4} + 7B$

Dash = $\frac{A+B^2}{A+\frac{1}{B}} + \frac{A}{B}$

Mash = $\frac{1}{\frac{1}{A}-\frac{1}{B}}$

Trash = $\frac{4}{\frac{A}{B}}\left[\frac{1+\frac{5}{C+D}}{A}\right]^{\frac{1}{2}} - \frac{A}{C+D}$

Input: An initial value N, followed by N sets of four values each for A, B, C and D (in that order).

Output: The values of the expressions Crash, Dash, Mash, and Trash (in that order). An error tolerance of +- 0.01 will be accepted. One blank line should separate each set of output.

Example Input File

```
3
12 4 3 1
8 4 9 3
```

Example Output to screen:

```
31.00
5.29
-6.00
-2.96

30.00
4.91
-8.00
-0.61
```

4. Day To Day

Program Name: daytoday.java Input File: daytoday.dat

General Statement : Given two dates, calculate and output the number of days between the two dates, exclusive. For example, there are 19 days between June 2, 2012 and June 22, 2012, namely June 3,4,5…21.

Input: An initial N indicating N sets of data to follow, each data set consisting of six integers, each set on one line, consisting of the starting month, day, and year and then the ending month, day, and year. The second date is guaranteed to be after the first date by at least one day, but could be in a later year.

Output: The number of days between each pair of given dates.

Example Input File

```
3
6 2 2012 6 22 2012
7 4 2011 12 25 2011
12 28 1980 1 1 1981
```

Example Output to screen:

```
19
173
3
```

5. Flex J Box

Program Name: flexjbox.java Input File: flexjbox.dat

General Statement : Given four positive integers, output the resulting "J in the box" as shown.

Input: An initial integer N, followed by N sets of four integers indicating rows, columns, and J position (row,col) in the box

Output: The resulting "J in the box" according to the given data, with at least one blank line of separation between each output. Note: The "J" will always be inside the box, and never on the border of the box.

Assumptions – Helpful Hints : Remember, row first, then column

Example Input File

```
2
4 5 2 2
10 9 3 4
```

Example Output to screen:

```
*****
*****
**J**
*****

*********
*********
*********
****J****
*********
*********
*********
*********
*********
*********
```

6. Gender Race

Program Name: gender_race.java Input File: gender_race.dat

General Statement : For the first time in the history of racing, males and females will compete against each other. What a concept! You have been hired to keep track of the winners. Given at most twenty names, finishing time and gender for each racer, determine the best and second best time overall, best and second best among the males, and finally best and second best among the females. There will always be at least two males and two females in the race.

Input: Integer N showing how many racers will be in the race, followed by N sets of data containing first name, gender, and finishing time for each racer. There will be no ties, and no more than 20 racers, and only one race.

Output: Name, time, and gender for each of the following:

- Overall winner
- Overall second
- Best Male
- Second place male
- Best female
- Second place female

Example Input File

```
6
Sue 48 F
Kelly 52 M
Stacey 43 F
Kelly 49 F
Stacey 50 M
John 44 M
```

Example Output to screen:

```
Stacey 43 F
John 44 M
John 44 M
Stacey 50 M
Stacey 43 F
Sue 48 F
```

7. Maze Map

Program Name: mazemap.java Input File: mazemap.dat

General Statement : When considering how to define a maze, two ways come to mind: list the "walls" or list the "interior passageways." In this problem, the "interior passageways" method is used. The mazes used will be square grids of N by N vertices. For input data, rather than listing all of the edges, the interior path is encoded using hex values as follows: `4 ECAE 844A.` This indicates a 4X4 maze, with `ECAE` representing the four rows starting at the top. The hex value E translates to the top row 1110, where a "1" indicates an edge or "pathway" between two vertices at the top left of the maze, and "0" the absence of an edge. Since it a 4X4 maze, there are only three possible passageways along the top row, therefore the last trailing"0" is extra and meaningless.

```
*-*-*-*
I     I
*-*-* *
  I I
*-* *-*
      I
*-*-*-*
```

The second row, indicated by the hex value "C", or bitstring 1100, means that there are two open passageways on the second row from left to right, but a wall between the 3rd and 4th vertices. Again, the final training zero is meaningless. Continue in this way until all rows are encoded.

The encoding of the vertical paths is similar, but consider the columns starting at the left and going to the right. Within each column, the bitstring again indicates vertical passageways that are open or closed. The value 8 is the bitstring "1000", which indicates an open vertical passageway at the top left, with no others in that first column. Again, the last zero in the string is meaningless.

For the input line `4 ECAE 844A`, the maze would look like this, where the "`*`" character represents vertices, the "`-`" horizontal passageways, and the "`I`" vertical passageways:

Your job is to output the bitstring representation of the horizontal passageways, and then of the vertical passageways. Do not include meaningless trailing bits, if any.

Input: An initial integer N, followed by N sets of data, each set consisting of an integer S and two hex strings. The first hex string represents the horizontal passageways of the SXS maze, the second the vertical passageways. For mazes larger than 5X5, two hex digits are required for each row and column, resulting in several possibly meaningless trailing zeroes. No mazes larger than 9X9 will be used in this problem.

Output: The resulting bitstring representations of the horizontal and vertical passageways as described above, with single spaces between each bitstring and one blank line separating each complete output.

Example Input file

```
2
4 ECAE 844A
6 68D8904098A0 C83018C83088
```

Output to screen:

```
111 110 101 111
100 010 010 101

01101 11011 10010 01000 10011 10100
11001 00110 00011 11001 00110 10001
```

8. Snapshot Day

Program Name: Snapshot.java Input File: snapshot.dat

General Statement : Schools track tardies and attendance every day, but once a year take a "snapshot" of the situation. Your job is to take the data gathered and do a brief analysis report.

Input: Data gathered for each of six classrooms, one each for classes 1 through 6, consists of total enrolled in the class, number absent that day, and number tardy. There will be no duplicate data sets.

Output: The output will be as follows:

1. Total number enrolled in all classes
2. Total number of tardies in all classes
3. The class with the best attendance (highest percentage not absent)
4. The list of classes in order from best to worst attendance for the day according to percentage not absent.

Example Input File

```
10 1 1
10 0 1
20 3 2
20 6 0
20 1 3
20 4 6
```

Example Output to screen:

```
100
13
2
2 5 1 3 6 4
```

9. Taxi, Taxi, UIL Taxi!

Program Name: taxi.java Input File: taxi.dat

General Statement : The cost to ride a taxi in UIL land is 50 cents for the first 1/5 mile (or less) and 22 cents for each additional 1/5 mile or part thereof. In addition, the taxi will wait for you while you conduct your business. The cost for waiting is 20 cents per 60 second period or any part thereof.

Input: Several sets of data, each set consisting of two non-negative values, each set on one line, separated by a space. The first value is the number of miles traveled, and the next is the number of seconds of waiting time.

Output: The cost of each taxi ride, with a +- 1 cent tolerance of error. Dollar format output required, each answer on one line.

Example Input File

```
1 0
2 0
1.5 75
3.4 125
```

Example Output to screen:

```
$1.38
$2.48
$2.44
$4.62
```

10. Test Parity

Program Name: testparity.java Input File: testparity.dat

General Statement : In a somewhat misguided effort to "level the testing field", your teacher has decided to assess a "tax" on all test scores. Tests will be greater than zero but no higher than 120. Using the following tax table, all test scores will be adjusted so it appears that everyone seems to be doing fairly well, hoping the administration will be pleased.

Score Range:	Action:
0 < score <= 25	Double the score
25 < score <= 50	Increase score by 50%
50 < score <= 75	Increase score by 25%
75 < score <= 100	No change
100 < score <= 120	Decrease score by 10%

Your job is to write a program that will make the appropriate adjustment for each score.

Input: Several test scores, all on one line, each separated by one space.

Output: The rounded integer adjusted scores for each original score, all on one line, each separated by at least one space.

Example Input File

```
12 50 80 117
```

Example Output to screen:

```
24 75 80 105
```

11. Weird Change

Program Name: WeirdChange.java Input File: weirdChange.dat

General Statement : In UIL land they do things weird, including their monetary system, where the basic unit is, you guessed it, the "uil"! Each "uil" is worth 3 cents in US currency. The only coins in this system have values of 1, 5, 13, 23, 37 and 47. Your job is to help your UIL coach with his money during the visit to UIL land. The reason he/she needs help is because of his/her obsession with giving exact change as well as the absolute fewest number of coins necessary for the transaction.

Input: Several positive integers, each between 1 and 200, representing how many "uils" something costs.

Output: The dollar amount equivalent to the given "uil" cost, and six values representing the exact change using the fewest "uil" coins (in descending order of value) needed to make the transaction. Assume your coach always has enough coins to make the purchase.

Example Input File

```
12
41
34
50
```

Example Output to screen:

```
$0.36 0 0 0 0 2 2
$1.23 0 0 1 1 1 0
$1.02 0 0 1 0 2 1
$1.50 0 1 0 1 0 0
```

12. Y Tree

Program Name: uiltree.java Input File: uiltree.dat

General Statement: Given a positive odd number greater than 2 but less than 10, output the resulting tree as shown below. For proper credit, the **output must be produced dynamically**, NOT by hardcoding.

Input: At least one and up to four odd integer values from the set {3,5,7,9}.

Output: A 'Y' shaped tree as shown below. The tree must be aligned on the left edge of the screen, and there is to be NO blank line between each tree output.eeeeeee

Example Input file

```
3 5
```

Output to screen:

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