A+ Computer Science February 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	Picture
Problem 2	Boxes
Problem 3	Backwards
Problem 4	Rocks
Problem 5	Triangles
Problem 6	Hoop
Problem 7	Quantum Fun
Problem 8	Physics Phun
Problem 9	Stock it to me
Problem 10	Wavey words
Problem 11	Skippy
Problem 12	Sum Digits

Good luck!

1. Picture

Program Name: Picture.java Input File: none

General Statement : Print out the picture as shown below.

Input: none

Output: Print out the picture as shown below.

Assumptions – Helpful Hints : none

Example Input File
none

Example Output to screen:
```
####################
##331234566543213 3##
#########33#########
##337777777777773 3##
#########33#########
#########33#########
##337777777777773 3##
#########33#########
##331234566543213 3##
####################
```

2. Boxes

Program Name: Boxes.java Input File: boxes.dat

General Statement : Print out each box as shown below.

Input: The first line in the data file will indicate the number of data sets to follow. Each data set will contain the size of the box to be printed.

Output: Print out each box of the appropriate size as shown below.

Assumptions – Helpful Hints : none

Example Input File

```
3
3
5
4
```

Example Output to screen:

```
$$$
$ $
$$$

$$$$$
$   $
$   $
$   $
$$$$$

$$$$
$  $
$  $
$$$$
```

3. Backwards

Program Name: Back.java Input File: back.dat

General Statement : Read in a string and determine if that string reads the same forwards as backwards.

Input: The first line in the data file will indicate the number of data sets to follow. Each data set will contain a single string with no spaces.

Output: Print out `SAME` if the word reads the same forwards as backwards or `NOT SAME` if the word does not read the same forwards as backwards.

Assumptions – Helpful Hints : none

Example Input File

```
3
mom
dog
pumpkin
```

Example Output to screen:

```
SAME
NOT SAME
NOT SAME
```

4. Comp Sci Rocks!!

Program Name: Rocks.java Input File: rocks.dat

General Statement : You think Comp Sci Rocks!! and you just like saying that over and over again. Read in a number from the data file and print out Comp Sci Rocks!! that number of times.

Input: The data file will contain an unknown number of lines.

Output: Print out the number of lines in the data file.

Assumptions – Helpful Hints : none

Example Input File

```
3
2
5
1
```

Example Output to screen:

```
Comp Sci Rocks!!
Comp Sci Rocks!!

Comp Sci Rocks!!
Comp Sci Rocks!!
Comp Sci Rocks!!
Comp Sci Rocks!!
Comp Sci Rocks!!

Comp Sci Rocks!!
```

5. Triangles

Program Name: Triangles.java   Input File: triangles.dat

General Statement : Read in a letter and a number. The number indicates how big the letter triangle should be. The number indicating the size of the triangle will have a range from 0 to 250. num>=0 and num<=250

Input: The first number indicates the number of data sets to follow. Each data set will contain one letter and one number. All letter input will be uppercase.

Output: Print out the appropriate sized letter triangle as shown below. The letter provided is the starting letter and that letter is printed once and then the sequence continues.

Assumptions – Helpful Hints : The letters must wrap around from Z to A. If you start with Z and have to print 5 levels, you must wrap around and start with A after the Z level is complete. The triangle lettering starts at the bottom and goes up.

Example Input File

```
3
5 A
3 Z
4 C
```

Example Output to screen:

```
EEEEE
 DDDD
  CCC
   BB
    A

BBB
 AA
  Z

FFFF
 EEE
  DD
   C
```

6. HoOp

Program Name: Hoop.java Input File: none

General Statement : Print out the word `Hoop` as shown below.

Input: none

Output: Print out the word `Hoop` as shown below.

Assumptions – Helpful Hints : none

Example Input File
none

Example Output to screen:
```
#  # #### #### ####
#  # #  # #  # #  #
#### #  # #  # ####
#  # #  # #  # #
#  # #### #### #
```

7. Quantum Fun

Program Name: Quantum.java Input File: quantum.dat

There are 4 quantum numbers (n, l, m_l, and m_s) and several rules (the Aufbau Principle, Hund's Rule, the Pauli Exclusion Principle) that we use to find electron configurations in Chemistry class. Electrons are arranged in s and p orbitals and those s and p electrons are called valence electrons. Only 2 electrons exist in the s orbital (opposite spin) but since there are 3 p-orbitals, 6 electrons can occupy the p sublevel.

Sulfur has 6 valence electrons in the 3rd main energy level (the 3rd row on the periodic table). So the electron configuration of S = [Ne] $3s^2p^4$ tells us that there are 2+4 = 6 electrons on the 3rd main energy level. If there are only 1 or 2 electrons, they do not fill the p-orbitals. Calcium, for example, Ca = [Ar] $4s^2$ is on the 4th main energy level, but only has 2 valence electrons (so there are no p-electrons).

Given the row and number of valence electrons, find the electron configuration of an element!

Input: The first line consists of the number of data set (the elements), followed by that number of lines. Each subsequent line contains 2 numbers, the row on the periodic table (N) and the number of valence electrons (V). V must be at least 1 and at most 8.

Output: Show the electron configuration in the form NsX or NsXpY, where N is the row on the periodic table, X is the number of s-electrons (1 or 2) and Y is the number of p-electrons (1-6).

Example Input File

```
4
4 2
3 6
5 8
1 1
```

Output to screen:

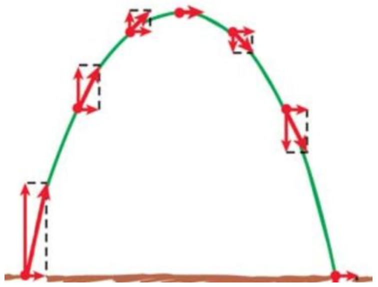
```
4s2
3s2p4
5s2p6
1s1
```

8. Physics Phun

Program Name: Physics.java Input File: physics.dat

When a projectile is launched at an angle with a certain velocity, we can predict the time and distance it will travel using several equations of motion. One of them will find the distance a projectile will travel given the initial velocity and the angle.

$$\Delta x = \frac{v^2 \sin 2\theta}{g}$$



V is the velocity (m/s), θ is the angle (in degrees), g is the acceleration due to gravity (9.81 m/s^2), and Δx is the horizontal distance traveled (m).

Note that an angle of 45^o provides the maximum distance for a given velocity, and complementary angles yield the same horizontal distance. You might need the documentation on the Math class and the methods dealing with trigonometric functions.

Input: The first line consists of the number of data sets in the file. Each subsequent line will have two values with decimals, the velocity and the angle.

Output: Show the distance rounded to the nearest mm (thousandth of a meter) in the form:
"x = ___ m".

Example Input File

```
15
20 20
20 25
20 30
20 35
20 40
20 44.8
20 45
20 45.2
20 50
20 60
20 70
20 90
50 45
50 25
50 65
```

(Continued on next page…)

(Problem 8 continued)

Output to screen:
```
x = 26.209 m
x = 31.235 m
x = 35.312 m
x = 38.316 m
x = 40.155 m
x = 40.774 m
x = 40.775 m
x = 40.774 m
x = 40.155 m
x = 35.312 m
x = 26.209 m
x = 0.000 m
x = 254.842 m
x = 195.220 m
x = 195.220 m
```

9. Stock It To Me

Program Name: Stock.java Input File: stock.dat

Many stock traders have tried to use data trends to predict the stock market to try to "make it big." You are employed by another such trading firm. Your job is to track a stock price for a month (20 trading days) and decide whether to buy, sell, or hold. You want to buy low at the start of a trend, and sell high at the end of a trend. Here are the determining factors:

- If the price goes up for 10 straight days, then sell that stock, predicting it will fall.
- If the price goes down for 10 straight days, then buy that stock, predicting it will rise.
- If the price goes up and down, but has a general increase of 20% from the initial price at 10 days ago, then sell that stock for a profit.
- If the price goes up and down, but has a general decrease of 20% from the initial price, then buy that stock hoping for a future increase.
- Otherwise, hold the stock until a trend develops.

Here is an example:

```
1.00 2.00 2.50 3.00 3.50 4.00 4.50 5.00 5.50 6.00 6.50 7.00 6.50 6.00 ...
```

For the previous trend, you should sell at 6.50 due to 10 straight days of increase.

Another example:

```
6.00 5.00 5.50 6.00 7.50 4.00 4.50 5.00 6.50 7.00 8.50 7.00 6.50 6.00 ...
```

For the previous trend, there was fluctuation up and down. On the 11th day, there was a 42% increase from the 1st day (from 6.00 to 8.50), so you should sell at 8.50.

To simplify this program assume:

- There can only one trend per month.
- The trend may not start immediately, but will start by the 11th number (the last 10 could be a trend).

Input: The first line will indicate the number of lines in the data file. Each subsequent line contains 20 stock prices.

Output: Show one of the following possible outcomes:

- `buy at __`
- `sell at __`
- `hold`

(Continued on next page…)

(Problem 9 continued)

Example Input file   (Note: in the actual file all 20 numbers are on a single line)

```
5
1.00 2.00 2.50 3.00 3.50 4.00 4.50 5.00 5.50 6.00 6.50 7.00 6.50 6.00 7.00 6.00 7.00 8.00
8.00 8.00
6.00 5.00 5.50 6.00 7.50 4.00 4.50 5.00 6.50 7.00 8.50 7.00 6.50 6.00 6.00
6.00 7.00 6.00 5.99 5.98
8.00 7.99 7.98 7.97 7.96 7.95 7.94 7.93 8.00 7.92 7.91 7.90 7.89 7.88 7.87
8.00 7.86 7.85 7.84 7.83
9.87 12.55 11.94 11.25 20.12 19.87 15.45 13.98 12.50 11.99 10.54 10.44 10.32
10.01 9.23 9.21 10.21 11.00 12.00 13.00
10.00 9.50 10.00 9.50 10.00 9.50 10.00 9.50 9.25 9.15 9.00 8.75 8.50 10.00 8.50 8.25 8.22
8.05 7.99 7.01
```

Output to screen:

```
sell at 6.50
sell at 8.50
hold
buy at 9.23
buy at 7.01
```

10. Wavey Words

Program Name: Wavey.java   Input File: wavey.dat

Take a word and print it out like a wave that goes up and down twice (2 wavelengths). Each word will be at most 10 characters and at least 3 characters.

Input: The first line contains the number of words in the data file. Each subsequent line contains a single word from 3 to 10 characters in length.

Output: Print the wave pattern as shown below, separating each wave by a blank line.

Example Input file

```
3
wave
groovy
wow
```

Output to screen:

```
w     w     w
 a   a a   a
  v v   v v
   e     e

g         g         g
 r       r r       r
  o     o   o     o
   o   o     o   o
    v v       v v
     y         y

w   w   w
 o o o o
  w   w
```

11. Skippy

Program Name: Skippy.java Input File: skippy.dat

When young students in elementary school learn math, they begin with their math facts. Skip counting is a method they learn to build their math skills to prepare them for multiplication. You want to write a program to help your younger sibling practice their “skip counting.” You will show the first twelve numbers of each skip count, counting by 2’s, 3’s, 5’s or other numbers less than 12.

Input: The first line consists of the number of data sets in the file. Each line consists of a single integer from 2 to 12.

Output: Print out the first 12 multiples of each number separated by a single space.

Example Input file

```
5
2
3
5
6
12
```

Output to screen:

```
2 4 6 8 10 12 14 16 18 20 22 24
3 6 9 12 15 18 21 24 27 30 33 36
5 10 15 20 25 30 35 40 45 50 55 60
6 12 18 24 30 36 42 48 54 60 66 72
12 24 36 48 60 72 84 96 108 120 132 144
```

12. Sum Digits

Program Name: Digits.java Input File: digits.dat

When young students in elementary school learn math, they begin with their math facts. Digit summing is another early math fact process they must learn. Write a program that will read in any number and sum up all of its base 10 digits. You might find % quite useful for this problem.

Input: The first line consists of the number of data sets in the file. Each line consists of a single integer.

Output: Print out the sum of all of the digits in each number.

Example Input file
```
5
100
88
123456
999
10101010101010101010
```

Output to screen:
```
1
16
21
27
10
```