PSCSTA

April 2013 –Packet 1

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
Problem 1	Almost Prime
Problem 2	KoolAid
Problem 3	Foreign Shoes
Problem 4	Taxation
Problem 5	Crossword Clues
Problem 6	Right Stuff
Problem 7	Periodic Permutations
Problem 8	Deer Tags
Problem 9	Vowely Words
Problem 10	Tiles
Problem 11	Bubble Pop 1
Problem 12	Bubble Pop 2

Good luck!

Updated for PSCSTA Programming Contest, April 2013

1. Almost Prime

Program Name: AlmostPrime.java Input File: almostprime.dat

Prime numbers have only two factors: one and itself. There are some numbers that have only three factors: one, another number and itself. For example,

the factors of 4 are 1, 2, 4 (1^{st} number is 4)
the factors of 9 are 1, 3, 9 (2^{nd} number is 9)
the factors of 25 are 1, 5, 25 (3^{rd} number is 25)

Find the first 40 numbers that are "almost prime!"

Input: There will be an unknown number of lines in the data file. Each line contains a single integer, N. The largest N will be 40.

Output: Print out the Nth number that is "almost prime."

Example Input File

```
1
2
3
7
```

Output to screen:

```
4
9
25
289
```

2. KoolAid

Program Name: KoolAid.java Input File: koolaid.dat

In the old days (when I was a kid), KoolAid packets made only enough for a 2 quart pitcher, and you had to add 1 cup sugar. Now they come in packets that you add to water bottles (in liters). You can make a 0.5 liter (small water bottle) or a 1.0 liter (medium water bottle) and a 2.0 liter (large water bottle) also. 2 people can share a 1.0 liter or 4 people could share a 2.0 liter bottle, but 5 people would need a 2.0 L and a 0.5 L bottle mixed. Given an amount of people, find the MINIMUM amount of KoolAid to make with NO left-over KoolAid. For example, you could make a 2.0 liter for 1 person, but a 0.5 liter bottle would be less wasteful.

Each person will drink 0.5 liters of Kool-Aid. Make enough for N people using large (2.0 L), medium (1.0 L) and small (0.5 L) bottles. Make only enough for N people with no wasted Kool-Aid.

Input: The first line consists of the number of data elements in the file, followed by that number of lines. Each subsequent line contains one integer, N, the number of people.

Output: Print the number and type of water bottle, starting with "large", then "medium", then "small".

Example Input File

```
6
1
2
3
4
20
99
```

Output to screen:

```
1 small
1 medium
1 medium 1 small
1 large
5 large
24 large 1 medium 1 small
```

3. Foreign Shoes

Program Name: ForeignShoes.java Input File: foreignshoes.dat

Shoe sizes in the US and the UK are close, but you want your shoes to fit well! If you are in the UK, find the US shoe size and vice-versa. Here are the conversions:

Women: UK 8 == US 10 (add 2 for US)
Men: UK 9 == US 10 (add 1 for US)

Input: The first line consists of the number of data sets in the file. Each data set will consist of 2 strings and an integer: the gender, the country, and the size (separated by a single space).

Output: Print out the gender, the country and the converted size (separated by a single space).

Example Input File

```
3
Women UK 7
Men US 10
Men UK 10
```

Output to screen:

```
Women US 9
Men UK 9
Men US 11
```

4. Taxation

Program Name: Taxation.java Input File: taxation.dat

When you check out at the grocery store, the receipt shows whether or not the item was taxable. Most grocery items are not taxed, but some supplies and "junkfood" items are taxed. If it is taxable, they add the tax only to those items. Given a receipt (indicating if it is taxable), calculate the total cost on the bill. Assume a tax rate of 8.25% on the taxable items, and do not round until the last sum.

Input: The first line consists of the number of items purchased. Each subsequent line contains the cost of the item, with a "T" at the beginning (followed by a space) if it is taxable.

Output: Show "The total is $___".

Example Input File

```
10
2.99
T 3.99
25.20
T 25.00
19.00
T 19.99
5.00
6.00
T 7.00
T 8.00
```

Output to screen:

```
The total is $127.45
```

5. Crossword Clues

Program Name: CrosswordClues.java Input File: crosswordclues.dat

You are writing crossword puzzles and you have started a new puzzle with some of the words filled in. Now, you are trying to find words that complete the partial letters going down or across. Given a partial word, see if it matches a word in your word bank. There may be more than one match, so print out all possible matches. Blank letters in the puzzle will be represented by an asterisk (*).

For example, you have a word with 3 letters starting with 'm'. Here are some matches for a three-letter word starting with m:
m**: "mom" , "man" , "moo" , "met" , "mat" , "men"…..

In this program, the longest word in the word bank will be 7 letters.

Input: The first 10 lines consist of 10 words each (the 100 words in the word bank—the judge data file will contain the same words). The rest of the file contains an undetermined number of partial words, each on one line.

Output: For each partial word, print out all the words that match (in the order they appear in the word bank) on one line. If there are no matching words, print out "NO MATCH."

Example Input file

```
as at aft and ant apps amble ample apple applet
be bean beat bend bump bunt bust butte battle before
ebb end eon east ends ever eves either esters eastern
fen fro fun fend fern from font fuzz fonts front
gin gun gut gins guns gust guts gusty gutsy goner
hat hit hot hut hats hits host huts horses hotter
pen pin pun port post pots punt porter potter punter
sap sip sop saps sips sops soot sort scoot skirt
tap tip top taps tips tops toot tort trips troop
zap zen zip zit zaps zips zits zebra zebras zipper
a*
z*
z**
a**le
f*n*
***t
```

Output to screen:

```
as at
NO MATCH
zap zen zip zit
amble ample apple
fend font
beat bunt bust east font gust host port post punt soot sort toot tort
```

6. Right Stuff

Program Name: RightStuff.java Input File: rightstuff.dat

In science class, you want to get the right answer. Scientists measure accuracy (how close you are to the accepted value) and precision (how close all your data values are together). You want to get the "right stuff."

In this program, take a data set and calculate the accuracy and precision. For the accuracy, compare the average to the true answer. The percent error needs to be less than or equal to 5% (plus or minus). For the precision, compare the range (high – low) with the average. The range must be less than or equal to 10% of the average of all values.

Input: The first line consists of the number of data sets in the file. Each subsequent line will contain a series of N double type variables. The first number is the accepted value, and the other (N-1) values make up the experimental data. N will be at least 5 and at most 50.

Output: Print out "Accurate", "Precise", "Both", or "Neither."

Example Input file

```
3
2.75 2.68 2.70 2.71 2.75 2.75 2.76
3.14 2.14 4.14 2.14 4.14 2.14 4.14 2.14 4.14
9.99 6.99 7.01 7.11 6.98 7.00 7.05 7.09
```

Output to screen:

```
Both
Accurate
Precise
```

7. Periodic Permutations

Program Name: Periodic2.java Input File: periodic2.dat

You can make the word "chocolate" from the symbols of elements on the periodic table:
"C", "H", "O", "C", "O", "La", "Te"
using the symbols Carbon, Hydrogen, Oxygen, Lanthanum and Tellurium. You can also use Cobalt for the "Co" before "La".

I also saw a UIL Science team t-shirt using the word "champions" from Carbon, Hydrogen, Americium, Phosphorus, Iodine, Oxygen, Nitrogen and Sulfur:
"C", "H", "Am", "P", "I", "O", "N", "S"

The word "banana" or "bananas" can be made with "Ba", "Na", "Na" and "S", but not with "N" because there is no "A" or "An" on the periodic table. Trying to use Nitrogen "N" would appear to fail when checking the rest of the word, but using Sodium "Na" would work for the rest. So you need to try any combination of 1 and 2 letter element symbols

In this program, determine if a lowercase word can be made with any combination of 1 or 2 letter element symbols.

Input: The first 4 lines will each contain 25 atomic symbols. The 5th line contains an integer N, representing the number of words to check. The next N lines each contains a single word (no spaces or special characters) made of lowercase letters that is no longer than 30 characters. The first four lines of data will be the same in the judge data file.

Output: Print "yes" or "no" for each word indicating if it can be made from the element symbols.

Example Input file

```
H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sc Ti V Cr Mn
Fe Co Ni Cu Zn Ga Ge As Se Br Kr Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn
Sb Te I Xe Cs Ba La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb Lu Hf Ta W Re
Os Ir Pt Au Hg Tl Pb Bi Po At Rn Fr Ra Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm
9
chocolate
rose
bananas
champions
north
south
uil
ladygaga
xenophobe
```

(Continued on next page…)

(Problem 7 contin.)

Output to screen:

```
yes
no
yes
yes
no
yes
no
yes
yes
```

8. Deer Tags

Program Name: DeerTags.java Input File: deertags.dat

You work for a wildlife biologist who is tracking deer with radio transmitter tags. The deer can be tracked over time to see if or how long they can survive. If a radio tag is moving, it is assumed the deer is still alive. If the radio tag is stationary for a day, the body is searched for. If a tagged carcass is found, the cause of death could be from natural causes or by a predator. Given the radio data, write a program tabulating the statistics for the deer population observed.

Each time a deer is observed, the radio id and a status report will be logged. If a live deer is found, it can be tranquilized and tagged. If a dead deer with a tag is found, the type of death will be logged.

In this program, each line of data will contain a 4 character string followed by a status report. The optional status reports are as follows:

- OK
- NEW
- DEAD – COYOTE
- DEAD – MOUNTAIN LION
- DEAD – BEAR
- DEAD – NATURAL CAUSES

Given a database of radio tag information, print a report with statistics of how many deer are still alive or dead (ranked by cause of death from highest to lowest). All non-dead dears are considered to be alive. If there is a missing category in a data set, show "0%" after the category.

Input: There are an unknown number of data sets in the file. Each data set consists of a radio id (4 character string) followed by a status report (from the above list).

Output: Print out statistics (nearest percent) for the number of deer still alive. Then rank the cause of death from highest to lowest, showing the percent of each type. If there is a missing category, show "0 %".

(Continued on next page…)

(Problem 8 contin.)

Example Input File

```
A001 NEW
A002 NEW
A001 OK
A003 NEW
A003 DEAD - NATURAL CAUSES
A004 NEW
A002 OK
A001 OK
A005 NEW
A006 NEW
A007 NEW
A008 NEW
A009 NEW
A010 NEW
A002 DEAD - NATURAL CAUSES
A010 DEAD - COYOTE
A009 DEAD - MOUNTAIN LION
A008 DEAD - MOUNTAIN LION
A007 DEAD - MOUNTAIN LION
A011 NEW
```

Output to screen:

```
ALIVE 45%
MOUNTAIN LION 27%
NATURAL CAUSES 18%
COYOTE 9%
BEAR 0%
```

---

9. Vowely Words

Program Name: Vowely.java Input File: vowley.dat

Some words have a lot of vowels (aeiou) compared to consonants. Given a word, determine if it is "vowely" or not. A word is "vowely" if at least half of its letters are vowels.

Input: The first line will indicate the number of words. Each line will contain one word (no spaces).

Output: Print out either "YES" or "NO".

Example Input file

```
6
banana
mystic
baaxuwaashee
regular
complex
regulate
```

Output to screen:

```
YES
NO
YES
NO
NO
YES
```

10. Tiles

Program Name: Tiles.java Input File: tiles.dat

You work for a company that lays floor tile. You run the website and want to put a "tile calculator" on the site. The customer will enter the length and width (in feet) of the rectangular area to be tiled, and you will tell them how many tiles (1 square foot each) they will need. When the tile layers talk to you, they tell you that there are some edges they have to cut -- so you will need to allow for 10% extra tiles for cutting.

If you calculate a fractional number of tiles, then make a whole number by using the next highest integer. For example, if you have a 14 by 16 foot room, the area is 224. Using 10% more, it would be 246.4 tiles: use 247 tiles.

Input: The first line will indicate the number of data sets. Each data set will consist of two integers, the length and width.

Output: Print out the number of tiles (an integer).

Example Input file

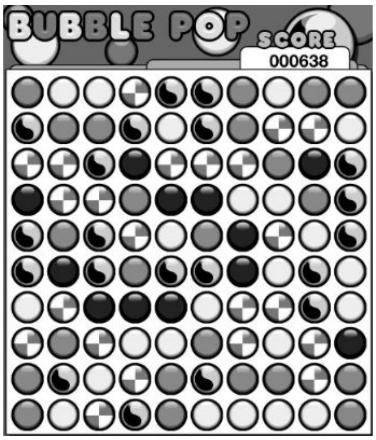
```
3
14 16
10 20
11 21
```

Output to screen:

```
247
220
255
```

11. Bubble Pop 1

Program Name: BubblePop1.java Input File: bubblepop1.dat



I have a couple of games on my Kindle, and Bubble Pop is one of my favorites. When you select contiguous bubbles (3 or more connected vertically or horizontally), they pop and the bubbles above them fall down in their place. You get a higher score for popping groups of bubbles that are larger. You keep popping bubbles until the largest contiguous bubble size is 2. There are 5 different "colors" of bubbles.

You are writing the code for this game! It is a complex program, so **you are writing only the first part - the check for contiguous groups of bubbles.** Given an array index of the matrix, determine if the bubble can burst and the number of the bubbles if it can. For this program, you will not be asked to burst the same bubble more than once (so you can alter the matrix and it will not affect later results).

Input: The first 10 lines consist of the game board, each line containing 10 capital letters (A-E) representing the different bubbles (the size of the game board is fixed at 10 by 10). The next line contains the number of data sets. Each data set contains two integers, the row and column of the bubble you are trying to pop. For example, "`0 0`" represents the character on the first line, "`A`" in the Example input file below.

Output: Print out either
"YES X" where X represents how large the group of bubbles is, or
"NO" if there are not at least 3 contiguous characters, connected horizontally or vertically.

Example Input file

```
ABBCDDABAA
DAADBDACCB
CCDECCCAED
ECCAEEBBAD
DADCBAECBD
DEDADDEBDB
BCEEEBCCDB
CACBBACBCB
ADBCADAACB
ABCDABBBBB
5
0 0
2 0
0 9
6 4
9 9
```

Output to screen:

```
NO
YES 4
NO
YES 3
YES 9
```

12. Bubble Pop 2

Program Name: BubblePop2.java Input File: bubblepop2.dat

This is a second program about the Bubble Pop game. In the real game, there is more complexity. When a bubble is burst, the cells from above fall down into the void created, and a random bubble is added to the top. For this program, always put the letter X at the top. Therefore, do not check to see if the letter X is contiguous to burst the bubble. Given a series of moves, determine if the game continues or if the game is over (when there is a maximum of 2 connected cells on the game board). If the game is over, assume the last move will cause this.

Note, that when one bubble is popped, the game board is altered. This may cause a new bubble to be created that can be popped. This is how you gain points in the game: keep popping new bubbles!

For example, when the 'C' at row 2 column 4 is selected, the three 'C's are deleted and the above cells fall down, leaving 3 'X's on the top row. Then, the 'D at row 1 column 3 could be a bubble with size 4:

```
ABBCDDABAA          ABBCXXXBAA          ABBXXXXBAA
DAADBDACCB          DAADDDACCB          DAACXXACCB
CCDECCCAED          CCDEBDAAED          CCDEBXAAED
ECCAEEBBAD          ECCAEEBBAD          ECCAEEBBAD
DADCBAECBD   --->   DADCBAECBD   --->   DADCBAECBD
DEDADDEBDB          DEDADDEBDB          DEDADDEBDB
BCEEEBCCDB          BCEEEBCCDB          BCEEEBCCDB
CACBBACBCB          CACBBACBCB          CACBBACBCB
ADBCADAACB          ADBCADAACB          ADBCADAACB
ABCDABBBBB          ABCDABBBBB          ABCDABBBBB
```

In this program, every selection will pop a bubble. The last selection could cause the game to end. If the maximum bubble size across the entire board is at most 2, then the game is lost.

(I have included an applet version of this game, so ask the contest director for the code AFTER the contest. It's not finished—a work in progress—but still kinda cool. :)

Input: The first 10 lines consist of the game board, each line containing 10 capital letters. The next line contains the number of data sets. Each data set contains two integers, the row and column of the bubble you are trying to pop.

Output: If the game ends, print out "GAME OVER". Then print out a blank line and the resulting game board.

(Continued on next page…)

(Problem 12 contin.)

Example Input file

```
ABBCDDABAA
DAADBDACCB
CCDECCCAED
ECCAEEBBAD
DADCBAECBD
DEDADDEBDB
BCEEEBCCDB
CACBBACBCB
ADBCADAACB
ABCDABBBBB
5
2 4
2 0
6 4
9 9
9 9
```

Output to screen:

```
XXXXXXXXXX
AXXCXXXBXX
DBBDDDACXX
EAAEBDAAAX
DADAEEBBCX
DEDCBAECEX
BCDADDEBAX
CACBBBCCBX
ADBCAACBCA
ABCDADAACB
```