A+ Computer Science October 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.

Number	Name
Problem 1	Part Of Speech
Problem 2	Touchy
Problem 3	Almost
Problem 4	Integral
Problem 5	War
Problem 6	Major MalFunction
Problem 7	Periodic Words
Problem 8	MixEmUp
Problem 9	Taxing
Problem 10	Speed
Problem 11	Resize
Problem 12	Puddle Bop

1. Part Of Speech

Program Name: PartOfSpeech.java Input File: partofspeech.dat

The word "die" has several meanings. Here are several:
die (v) – to stop living – He will die if he does not get a transplant.
die (v) – to stop working – The car can die if you get water in the fuel tank.
die (n) – a polyhedron used in games of chance – We roll a die in our game.
die (n) – a stamping or pressing tool – A die was placed in the machine to create our template.

There are many parts of speech in the English language. There are nouns, verbs, adjectives, adverbs, pronouns, conjunctions, definite and indefinite articles, and probably many more I would know if I were an English teacher.

In this program, you will determine if a word is a noun or verb based on the following criteria:

- If a word is preceded by the articles [a, an, the, this], it is a noun
- If a word is preceded by a helping verb [have, has, had, do, did, does…..], then it is a verb.

Input: The first line consists of the articles to determine if the part of speech is a noun. The next two lines contains of a list of helping verbs to determine if the part of speech is a verb. The next line consists of the number of data sets. Each data set consists of a word to search for and the number of sentences in that data set (N). There will be N lines with a sentence on each line. The first 3 lines of input will be identical to the judge data set!

Output: Print out the word for each data set followed by N lines showing either "NOUN" or "VERB" for each sentence in the data set. Separate data sets with a blank line.

Example Input File

```
a an the this
have has had do did does would could should can may might must
will shall am are is was were be been being
2
die 4
He will die if he does not get a transplant.
The car can die if you get water in the fuel tank.
We will roll a die to move the car in Monopoly.
A die was placed in the machine to create our template.
face 2
I can face my fears.
Please don't hit me in the face!
```

(Continued on next page…)

(Problem 1 contin.)

Output to screen:

```
VERB
VERB
NOUN
NOUN

VERB
NOUN
```

2. Touchy

Program Name: Touchy.java Input File: touchy.dat

I have a touch lamp that sits beside my bed. If it is off and you touch it, it turns on dim. If you touch it again, it turns on medium power. If you touch it again, it turns on high power. One more touch and it goes off. So if it is on low power, it takes 3 touches to turn it off. Since it is usually on low power before bedtime, I usually touch it 3 times to turn it off. However, if it's on medium and I touch it 3 times, it turns back on low!

Write a program that determines the final power setting of a lamp after it is touched a certain number of times! The lamp can have one of 4 power settings:

0123 (0=off, 1=low, 2=medium, 3=high)

Input: The first line consists of the number of data elements in the file, followed by that number of lines. Each subsequent line contains two integers: the initial power setting (0-3) and the number of times the lamp is touched.

Output: Print the final power setting for each data set (0-3).

Example Input File

```
6
0 1
1 2
2 3
0 20
3 22
3 7
```

Output to screen:

```
1
3
1
0
1
2
```

3. Almost

Program Name: Almost.java Input File: almost.dat

My daughter in kindergarten likes to play tic-tac-toe with me (at restaurants on the kid menu, or at home on paper, or in chalk on the sidewalk…). Her strategy is not high level yet, but she can always win if she has two letters in a row with one missing space. Write a program to make sure she always wins!

Input: The first line consists of the number of data sets in the file. Each data set will consist of 3 lines of 3 characters each (space, capital X, or capital O).

Output: For each data set, print out the row and column (1, 2, or 3). Every data set will have only one way for the X to win.

Example Input File

```
3
XOO
 X
 O
OXO
 X
  O

OXO
  X
```

Output to screen:

```
3 3
3 2
1 1
```

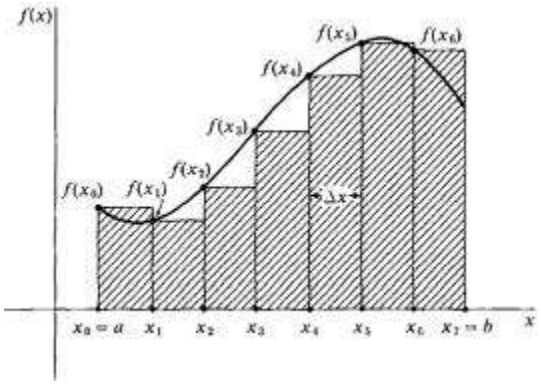
4. Integral

Program Name: Integral.java Input File: integral.dat

In calculus, the integral is the area under a curve (also the inverse of a derivative). In this program you will find a numerical value for the value of:

$$F(x) = \int_{x1}^{x2} f(x)dx$$

This is the area under the curve f(x) from x1 to x2 and is called a definite integral.



In a sense, you find the area of a bunch of rectangles and add them together! (For the calculus student, this is a left Riemann sum).

You will have an initial x, a final x, and how large dx will be. For this program, you will use the following function: $f(x) = 3x^2 + 2x + 1$.

Input: The first line consists of the number of data sets in the file. Each subsequent line will have 3 double variables: the initial x, a final x, and dx (the width of the rectangles).

Output: For each data set, print out the integral to six decimal places.

Example Input File

```
3
0 2.5 0.1
0 10 1.0
0 10 0.05
```

Output to screen:

```
23.200000
955.000000
1102.012500
```

5. War

Program Name: War.java Input File: war.dat

"All's fair in love and war." // old proverb

"War… what it is good for? Absolutely nothing!" // Edwin Starr

"There never was a good war, or a bad peace." // Benjamin Franklin

The only "war" that can be good is the card game☺. Write a program that simulates this card game. Here are the rules. Two (or more) players are dealt an equal number of cards in their pile. At each turn, players put one card face up. The person with the higher value card collects the cards when they win the hand. Here are the values lowest to highest: 2-10, J, Q, K, A. If there is a tie, then each player puts down 3 cards face down, and then replays the tie with the next card. The winner of the "war" gets all ten cards in the tie-breaker!

In this program, 2 players will have an equal number of 15 cards. The winner of each hand will collect the cards, so keep a running total for each player. If there is a tie (a war), then there will always be enough cards left to break the tie (at least 4 more cards will be in the data set). In this program, there will not be a double war (the 4 card after the tie will not be the same).

Input: The first line consists of the number of data sets in the file. Each data set contains two lines, each line containing 15 cards (2-10, J,Q,K,A).

Output: For each data set, print out the number of cards for each player on the same line separated by a space.

Example Input file

```
2
A K Q J 9 A K Q J 9 A K Q J 9
2 3 4 5 6 2 3 4 5 6 2 3 4 5 6
A K 10 5 J 9 A 4 5 6 2 3 4 5 Q
10 A 4 5 6 2 3 6 K Q J 9 A K J
```

Output to screen:

```
30 0
6 24
```

6. Major MalFunction

Program Name: MalFunction.java Input File: malfunction.dat

You are going to evaluate a quadratic function in the form $f(x) = ax^2+bx+c$, where a, b, and c are integers and x is a rational number (double variable).

Input: The first line consists of the number of data sets in the file. Each subsequent line will have 3 integer variables and 1 double variable: the coefficients of the quadratic polynomial and the independent variable, x.

Output: Print out the function value for each data set rounded to the nearest thousandth.

Example Input file

```
3
1 1 1 1
1 1 1 0.5
5 4 3 2
```

Output to screen:

```
3.000
1.750
31.000
```

7. Periodic Words

Program Name: Periodic.java Input File: periodic.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.

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"

Determine if a word can be made from the element symbols from the 1st 100 elements. In the word "chocolate" the underlined letters can be made from Co-Cobalt in addition to CO from Carbon and Oxygen. However, to simplify for this program, first check the single letter elements, and then check for the 2 letter elements. If the L is not able to make the word, then check for La in Lanthanum.

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
7
chocolate
rose
grass
champions
north
south
uil
```

Output to screen:

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

8. MixEmUp

Program Name: MixEmUp.java Input File: mixemup.dat

After playing the card game war (problem 5) you have some cards in piles. To play again, you have to shuffle the cards. Sometimes when you do not shuffle very well, there are still like cards together in the pile. If there are 2 adjacent cards, put one of these cards at the end.

For example, here is a list of cards:

2 5 8 K K 10 9 5 7 7 8

Move the K and the 7 to the end of the list:

2 5 8 K 10 9 5 7 8 K 7

For this program, there will at most 2 adjacent cards.

Input: The first line consists of the number of data sets in the file. Each data set consists of a series of up to 20 cards (2-10, J, Q, K, A) separated by a single space.

Output: Print out the modified list of cards for each data set.

Example Input file

```
3
2 5 8 K K 10 9 5 7 7 8
A K Q J 10 9 8 7 6 5 4 3 2
2 2 3 3 4 4 5 5 6 6 7 7 8 8
```

Output to screen:

```
2 5 8 K 10 9 5 7 8 K 7
A K Q J 10 9 8 7 6 5 4 3 2
2 3 4 5 6 7 8 2 3 4 5 6 7 8
```

9. Taxing

Program Name: Taxing.java Input File: taxing.dat

Amazon.com purchases used to have no sales tax in Texas. In addition to its free shipping, this was a great incentive to buy from them. Then it built warehouses in Texas, so sales taxes now apply to purchases for Texas residents. Use a sales tax rate of 8.25% and calculate the total bill for your order!

Input: The first line will indicate the number of data sets. Each data set will consist of a double variable (the purchase amount) on a separate line.

Output: Print out the total bill (with tax) with dollar sign in front and rounded to the nearest cent.

Example Input file

```
3
100.00
59.99
10.01
```

Output to screen:

```
$108.25
$64.94
$10.84
```

10. Speed

Program Name: Speed.java Input File: speed.dat

In physics class, you learn that the speed of sound is really slow compared to the speed of light. A good way to estimate distances is to measure how long it takes a sound wave to travel a certain distance. A good rule of thumb is to count the number of seconds when you see lightning or fireworks or other things that you can both see and hear. Sound can travel a mile in roughly 5 seconds. Given a time in seconds, estimate the distance in miles!

(To be more precise for physics class, the speed of sound at normal temperature and pressure is 343.0 m/s. In 5.000 seconds sound can travel 1715 m, which is 1.066 miles.)

Input: The first line will indicate the number of data sets. Each data set will consist of an integer, the time in seconds.

Output: Print out the distance in miles (rounded to the nearest tenth).

Example Input file

```
3
1
5
12
```

Output to screen:

```
0.2
1.0
2.4
```

11. Resize

Program Name: Resize.java Input File: resize.dat

When you resize a scanned/digital picture (larger or smaller) you don't always get the same ratio as the next photo paper size. For example, when you take a 5"x7" scan and try to resize to a 3"x5" the ratios are a little off. To the nearest hundredth, you can get either a 3.00" by 4.20" or a 3.57" by 5.00" using the 5:7 ratio of the original. Given a picture scan and a desired output, show both possible sizes so you know which way to crop!

Input: The first line consists of the number of data elements in the file, followed a picture size and the desired scaled picture size, each on one line. The input sizes will be integers.

Output: Show the two resized possibilities, to the nearest hundredth of an inch.

Example Input file

```
3
5 by 7 to 3 by 5
8 by 10 to 5 by 7
8 by 10 to 4 by 6
```

Output to screen:

```
3.00 by 4.20 or 3.57 by 5.00
5.00 by 6.25 or 5.60 by 7.00
4.00 by 5.00 or 4.80 by 6.00
```

12. Puddle Bop

Program Name: Puddle.java Input File: puddle.dat

You are playing a game where you are trying to jump over puddles of water. The game is set up as a matrix (rows and columns). You can only jump on small islands of dry ground. The water cells are labeled as "W" and the dry ground is labeled "G". There are rocks, "R", and sandy areas, "S". Once you jump over the water, is it frozen, so it turns to "F."

- If you jump onto a patch of dry ground (any size of contiguous G cells vertically or horizontally), then output "OK".
- If you jump onto water (any size of contiguous W cells), then that puddle turns into F cells and output "PUDDLE".
- If you jump onto rocks or sand (R or S), then output "OOPS".

Write the code for this game! Assume that the game either continues (G or W) or will end (R or S) on the last move.

Input: The first line will indicate the size of the game board (R and C both less than 20). The next R lines contain the matrix consisting of capital letters. The next line will contain the number of moves in the game, N (no more than 20). The next N lines will contain two integers, the row and column (array indices) of the move separated by a space.

Output: For each move, output the appropriate statement ("OK", or "PUDDLE", or "OOPS"). Then print out a blank line and the resulting game board.

Example Input file

```
10 10
GWGGGGRRGG
WWGGGGGSSG
RSGGGGSSSS
SSSWWWGGGG
SSRWWGGGGG
RRRWGGGGGG
GGGGRRRGGG
GGGSSSSSSG
WWWGGWWWWR
GWWWWWGGSS
6
0 0
1 1
2 2
2 5
4 4
9 9
```

(Continued on next page…)

(Problem 12 contin.)

Output to screen:

```
OK
PUDDLE
OK
OK
PUDDLE
OOPS

GFGGGGRRGG
FFGGGGGSSG
RSGGGGSSSS
SSSFFFGGGG
SSRFFGGGGG
RRRFGGGGGG
GGGGRRRGGG
GGGSSSSSSG
WWWGGWWWWR
GWWWWWGGSS
```