A+ Computer Science October 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
Problem 1	McSpeak
Problem 2	Are We There Yet?
Problem 3	Too Shy
Problem 4	Satisfied
Problem 5	Talk A Lot
Problem 6	Cool Colors
Problem 7	High And Low
Problem 8	Debt Is Dumb
Problem 9	Check Your Work
Problem 10	Note Interception
Problem 11	Personalized Plates
Problem 12	Save Like No One Else

Good luck!

1. McSpeak

Program Name: McSpeak.java Input File: mcspeak.dat

McDonald's food item often contain the letters "Mc" in front of the real name. For example, Nuggets become "McNuggets" and a Double becomes "McDouble." Add the letters "Mc" in front of the item!

Input: The first line consists of the number of data elements in the file (the number of food items), followed by that number of lines. Each line will contain a single-word food item.

Output: Show each food item with "Mc" before it.

Example Input File

```
3
Salad
Nuggets
Chicken
```

Output to screen:

```
McSalad
McNuggets
McChicken
```

2. Are We There Yet?

Program Name: ThereYet.java Input File: thereyet.dat

I rarely travel without my wife and children (except for UIL trips). In all my experiences travelling with my family, it takes longer to get the female members of my family out of the house and in the car. We also take restroom breaks on trips. So, for each trip length in hours and minutes, add 30 minutes to the trip!

Input: The first line consists of the number of data elements in the file (the number of trips), followed by that number of lines. Each subsequent line contains the hours and minutes of the trip in the form "X hours Y minutes", where X<24 and 0<=Y<60.

Output: Show the updated trip time in hours and minutes in the same form "`X hours and Y minutes`".

Example Input File

```
4
2 hours 20 minutes
3 hours 45 minutes
12 hours 0 minutes
2 hours 30 minutes
```

Output to screen:

```
2 hours and 50 minutes
4 hours and 15 minutes
12 hours and 30 minutes
3 hours and 0 minutes
```

3. Too Shy

Program Name: Shy.java Input File: shy.dat

Some people sitting in a row of seats (movie theater, lecture hall, airplane, etc.) do not like to be crowded. If they sit next to people, they like to be at the end seat. If they sit in the middle, they like to have a free seat next to them. So, given a row of seats, find out where they can sit!

Input: The first line consists of the number of lines in the file. Each subsequent line will have a string of all lowercase characters. The letter ‘x’ represents another person in that seat and the letter ‘f’ represents a free space. Each line will have a length of 5-20 letters.

Output: Use the letter ‘o’ to indicate an appropriate “open” space, so change an available ‘f’ to ‘o’. An available open space is either
a) free at the beginning or end of the string, or
b) is free in the middle of 2 other free spaces

Example Input File
```
5
ffffffff
fxxfffxxffff
fxffxfffxxxff
xxxxxxxxxxxxxxx
xfffxxxxfxxfxxf
```

Output to screen:
```
oooooooo
oxxfofxxfooo
oxffxfofxxxfo
xxxxxxxxxxxxxxx
xfofxxxxfxxfxxo
```

4. Satisfied

Program Name: Satisfied.java Input File: satisfied.dat

My wife does not like peanut M&M’s but I do. So of course we buy the plain M&M’s and I just buy some peanuts too. I like to pour them in a Dixie cup and snack away! But I only get “satisfied” when I eat equal number of M&M’s and peanuts, so it is not too salty and not too sweet.

I also like Snickers but that doesn’t make a good programming problem! So, given an amount of peanuts and M&M’s consumed, determine if I get satisfied! If I don’t get satisfied, determine how many of the lesser item I need to eat to become satisfied.

This is not a fictional program. I am eating them as I type this problem set! Crunch, crunch…

Input: The first line consists of the number of lines in the file. Each subsequent line will have a string of all lowercase characters. The letter ‘p’ represents a peanut and the letter ‘m’ represents an M&M. Each line will have a length less than 70 letters.

Output: Show the word “Satisfied!” if there are equal peanuts and M&M’s.
If not, show either “Eat __ more peanuts.” or “Eat __ more M&M’s”.

Example Input File

```
3
ppppmmmp
pppmmmpmmpmp
mmmmmmmmmmmmmmmmmmmm
```

Output to screen:

```
Eat 2 more M&M’s.
Satisfied!
Eat 20 more peanuts.
```

5. Talk A Lot

Program Name: Talk.java Input File: talk.dat

Some students can work well sitting next to anyone without being distracted. However, when some students sit by a certain other person, they cannot concentrate to maximize learning, which is what everyone wants. ☺

Our school's gradebook/attendance program will generate random seating charts for each class roster. Given a seating chart, determine if certain pairs of students are seated adjacent (either horizontally, vertically, or diagonally). To simplify the process, we will use the capital letters A-Z and digits 0-9 to represent student names for a maximum size of 36 in the class seating chart.

Input: The first line consists of the number of data sets in the file. The first line in each data set contains the number of rows (X), the number of columns (Y), and the number of student pairs to check for (Z). The next X lines contain the seating chart and the next Z lines contain the pairs to students to check for. Each seating chart will be rectangular, no more than 6 by 6 and no less than 2 by 2.

Output: For each seating chart, print "OK" if all the pairs are not adjacent. If a pair of students is adjacent (either vertically, horizontally, or diagonally) print out that pair. Print a blank line between different seating charts.

Example Input file
```
2
3 3 3
ABC
DEF
G89
AC
DC
G9
5 5 3
ABCDE
FGHIJ
KLMNO
56789
PQRST
GB
GJ
7N
```

Output to screen:
```
OK

GB
7N
```

6. Cool Colors

Program Name: CoolColors.java Input File: coolcolors.dat

My daughters' crayon box this year advertised that you can take your photos and create coloring pages with them. To make the black-line color pages, the photos must be scanned in some way, and then converted to take out the color. You will take a matrix of letters and replace all but the black colors, represented by the lowercase 'b'.

Input: There will be an unknown number of lines in the data file. Each line will contain capital and lowercase letters representing colors. Each line will be less than 70 letters.

Output: Print out the number of lines, replacing all letters but lowercase 'b' with a space ' '.

Example Input file
```
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
brrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrb
brrrrrrrrrrrrrrrrrbrrrrrrrrrrrrrrrrrrrrb
brrrrrrrrrrrrrrrrbBbrrrrrrrrrrrrrrrrrrrb
brrrrrrrrrrrbbbbbBBBbbbbbrrrrrrrrrrrrrrb
brrrrrrrrrrrrbBBBBBBBBBbrrrrrrrrrrrrrrrb
brrrrrrrrrrrrrbBBbbbBBbrrrrrrrrrrrrrrrrb
brrrrrrrrrrrrbBBBBBBBBBbrrrrrrrrrrrrrrrb
brrrrrrrrrrrbbbbbbbbbbbbbrrrrrrrrrrrrrrb
brrrrrrrrrrrrrrrrbBbrrrrrrrrbrbrbrbrrrrb
brrrrrrrrrrrrrrrrbBbrrrrrrrrbrbrbrbrrrrb
brrrrrrrrrrrrrrrrbbbrrrrrrrrbbbrbrbbbrrb
brrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

Output to screen:
```
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
b                                      b
b                 b                    b
b                b b                   b
b           bbbbb   bbbbb              b
b            b         b               b
b             b  bbb  b                b
b            b         b               b
b           bbbbbbbbbbbbb              b
b                b b        b b b b    b
b                b b        b b b b    b
b                bbb        bbb b bbb  b
b                                      b
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

7. High And Low

Program Name: HighAndLow.java Input File: highandlow.dat

You work for the Weather Channel website, and they need a program that will collect temperature data from a device every hour (along with other weather data). Given a list of 24 daily temperatures, find the high and low temperature for the day.

Input: The first line will indicate the number of lines in the data file. Each subsequent line contains 24 integers, the temperatures.

Output: Show the high and low temperatures in the form:
“`high = X low = Y`”

Example Input file

```
2
-5 -1 0 5 10 15 20 30 30 30 30 40 40 40 40 40 45 40 35 30 25 20 15 10
76 76 76 76 78 80 90 95 95 98 99 99 99 100 105 98 88 82 80 80 78 78 78 72
```

Output to screen:

```
high = 45 low = -5
high = 105 low = 72
```

8. Debt Is Dumb

Program Name: Debt.java Input File: debt.dat

Dave Ramsey is a financial counselor who teaches that “Debt is dumb, cash is king.” So he does not recommend borrowing money for a car. All cars go down in value with time. So how can you drive a car at all you may ask? Buy a car that does not go down in value too fast—buy a used car and pay cash for it! To determine whether a car fits in the Dave Ramsey plan, find the car value the year before and see if it decreased 10% or more in value. He calls this type of expensive debt-ridden car a “fleece” and the car that does not go down in value a “beater.”

See www.daveramsey.com for more details!

Input: There will be an undetermined number of lines in the data file. Each line contains two monetary values, the previous years’ value and the current value.

Output: Print out either “beater” or “fleece” for each line of data.

Example Input file
```
15000.00 10000.00
15000.00 13500.00
3000.00 2000.00
2999.99 2850.00
1.00 0.95
```

Output to screen:
```
fleece
fleece
fleece
beater
beater
```

9. Check Your Work

Program Name: Check.java   Input File: check.dat

My daughter in 4th grade this year will learn to multiply 2 and 3 digit numbers. What a great program! I have written a program myself that creates a random 2 digit multiplication problem worksheet written to 2 data files, one quiz and one key. Your mission, should you choose to accept it, is to show the work to the problem. For example, 45 x 32 = 1440. So show the ones digit product (45x2=90), the tens digit product (45x30=1350) and then the sum of those two numbers in the following format:

```
  45
 x32
---- (use 4 minus signs for the line)
  90
1350
----
1440
```

Input: The first line will indicate the number of data sets. Each data set will consist of 2 lines showing a multiplication problem of two 2-digit numbers. Assume the largest product will be a 4-digit number.

Output: Show the multiplication problem as shown above, each one separated by a blank line.

Example Input file

```
3
 45
x32
 99
x99
 11
x11
```

(Continued on next page…)

(Problem 9 contin.)

Output to screen:

```
  45
 x32
----
  90
1350
----
1440

  99
 x99
----
 891
8910
----
9801

  11
 x11
----
  11
 110
----
 121
```

10. Note Interception

Program Name: Note.java Input File: note.dat

In the days before texting, students had to pass notes written on paper around the classroom without the teacher noticing. Now you just wear a hoodie and text in the pocket, but none of you excellent, conscientious students would ever do such a thing! ☺

In this program, you will be given a seating chart where girls are capital letters and boys are lowercase letters. Only students of the same gender can pass and receive a note, so boys cannot pass to girls and vice-versa. Notes can only be passed to the person in front, behind, or to the side. It may or may not be possible to pass the note from the sender to receiver (they have to wait until after class), and it is possible that the person the note is written about might intercept and read the note (awkward moment). So, determine if the sent note's path is impossible, possible, or intercepted.

A path is impossible if there is no path available between sender and receiver (they are not contiguous). A path is possible if there is at least 1 valid path, and there is no chance for the note to pass through the subject of the note (sender and receiver are contiguous, but not with whom the note is about). A note is considered to be intercepted if the note's subject is contiguous with both the sender and receiver. It doesn't have to be directly in between the sender and receiver to be intercepted.

Input: The first line contains 2 numbers, R and C, the number of rows and columns of the seating chart matrix. The smallest matrix size is 9 and the largest is 52. The next R lines contain the seating chart. The next line contains an integer X, the number of data sets (notes to be sent). Each subsequent line is a string of 3 characters, representing the note's sender, then receiver, then who the note is about.

Output: There will be X lines of output, each line containing "POSSIBLE" or "IMPOSSIBLE" or "INTERCEPTED".

(Continued on next page…)

(Problem 10 contin.)

Example Input file

```
7 7
AaBbczy
fCZMNPe
gGHIJhi
jDkEFdm
KlLOoQq
rRSTUVs
WXtuvwx
5
tbm
tbf
tac
WQE
BFA
```

Output to screen:

```
INTERCEPTED
POSSIBLE
IMPOSSIBLE
INTERCEPTED
POSSIBLE
```

11. Personalized Plates

Program Name: Plates.java Input File: plates.dat

Some states issue personalized license plates for an extra charge. The car owners like to put funny and witty phrases, but they have to condense it to what fits on a license plate. Given a saying, convert it to a personalized license plate. I think it is fun trying to decipher these when you see them traveling on the highways.

Here are the guidelines:

1. A license plate saying can only be at most 8 characters long, no spaces. If it is 8 characters or less, it is already a valid license plate saying.
2. If it is longer than 8 characters, remove the first vowel in each word as you go through the words in the phrase. If still too long, continue by removing the next remaining vowel from each word in the phase. Repeat vowel removal until the saying is 8 characters long.
3. If all the vowels are taken out and it is still more than 8 characters, use only the first 8 consonants.
4. Vowels are considered to be "aeiou" only for this program.

For example, in the saying "great driver," first remove the space, then "e" in great, then remove the "i" in driver. The term "gratdrver" is still too long, so remove the "a" and it fits!

Input: The first line contains the number of lines of data in the file. Each subsequent line will contain a saying consisting one or more lowercase words.

Output: Show the 8 character or less, no space string for each saying.

Example Input file

```
6
great driver
funny man
cutie
very handsome doctor
voldemort
frodo lives
```

Output to screen:

```
grtdrver
funnyman
cutie
vryhndsm
vldemort
frdolves
```

12. Save Like No One Else

Program Name: Save.java Input File: save.dat

Another Dave Ramsey plan idea is to "Live like no one else, so later, you can live like no one else." In other words, sacrifice and save now to build a large nest egg for your kids' college savings and retirement. Compound interest is a powerful wealth building tool, so if you save a certain monthly amount for years, it will grow greatly beyond the amount invested. For example, if you save $200 a month for 20 years at 10% annual return, that $48,000 would grow to $151,873.77. Here is the formula for the balance of a periodic investment used by many financial website "calculators":

$$B = \frac{P\,[\,(1 + r)^n - 1]}{r}$$

where B is the balance after the investment period, P is the monthly investment, n is the number of months, and r is the monthly interest rate (annual / 12 so 6% would be 0.06/12 = 0.005). Given P, n and r, determine B, the balance at the end of the time period. Be sure to round only at the end.

Input: The first line will indicate the number of data sets. There will be 3 numbers (P, n, and r) on each line separated by a space.

Output: Output the total balance, showing commas and a dollar sign, rounded to the nearest penny.

Example Input file
```
4
200.00 240 10.0
200.00 360 10.0
400.00 360 10.0
400.00 276 12.0
```

Output to screen:
```
$151,873.77
$452,097.58
$904,195.17
$583,389.03
```