**Java program:** Prob01.java

**Input File:** Prob01.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

The ability to read a file is an essential tool in every programmer’s repertoire. Your task is to demonstrate that you’ve mastered this ability by reading the contents of a file and writing the file’s contents to the screen.

**Program Input**

The file Prob01.in.txt will contain some number of lines of text. The length of the lines can vary, but no line will be more than 100,000 characters long.

**Example Input:**

CodeQuestis awesome.

Somet!im@es# I l%ik^e&to\* u(se) e1xt$ra punctuation OR CHARACTERS.

**Program Output**

Your program should print out the input file exactly as it is read.

**Example Output:**

CodeQuest is awesome.

Somet!im@es# I l%ik^e&to\* u(se) e1xt$ra punctuation OR CHARACTERS.

**Java program:** Prob02.java

**Input File:** Prob02.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

The evil queen looks in her magic mirror to see things her way. While you can’t print out text the way it would really look in a mirror, you can reflect the order of a string. Your mission, should you choose to accept it, is to reverse a set of strings.

**Program Input**

The file Prob02.in.txt will contain some number of strings, one per line.

**Example Input:**

Who is the fairest of them all?

Now is the time to come to the aid of your country.

**Program Output**

Your program should output the strings in the same order, but reversed.

**Example Output:**

?llamehtfotseriafehtsiohW

.yrtnuocruoyfodiaehtotemocot emit ehtsiwoN

**Java program:** Prob03.java

**Input File:** Prob03.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Often, when working with files, it’s helpful to separate the extension from the filename. This is useful when wanting to perform an action on certain type of file. An example would when you want to move or rename all text files. Your task is to take a list of filenames as input and create a list of extensions along with their count.

**Program Input**

The file Prob03.in.txt will contain a list of filenames with extensions. Each filename will be on its own line.

**Example Input:**

File1.txt

File2.doc

File3.out

File4.txt

File5.so

File6.dll

**Program Output**

The output of the program should be the list of file extensions without the dot, followed by the number of times the extension was found. A single space should be between the extension and the count. The extensions will each be on separate lines and should only be listed once. List the file extensions in the order that you encounter them in the input file.

**Example Output:**

txt 2

doc 1

out 1

so 1

dll 1

**Java program:** Prob04.java

**Input File:** Prob04.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

In this exercise you will simply sort a listof integers.

**Program Input**

The file Prob04.in.txt will containa comma delimited list of integers, one set per line. You should sort each line of the input file separately.

**Example Input:**

1,5,8,6,7,4,8,5,23,45,85,15476,3548,1,5,2,457

5,4,75,6

8,7,5,987,4815,6873,548,3651536,53

**Program Output**

Your program should print out each line of numbers from least to greatest separated by commas with no spaces.

**Example Output:**

1,1,2,4,5,5,5,6,7,8,8,23,45,85,457,3548,15476

4,5,6,75

5,7,8,53,548,987,4815,6873,3651536

**Java program:** Prob05.java

**Input File:** Prob05.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Susan works for Widgets and Such in the IT Department. Part of Susan’s job is to remove old software from computers. Susan needs to run a program that can determine from program access logs which pieces of software have not been accessed within the last 180 days. If the software was accessed then it should remain on the system; otherwise that software is eligible to be removed. If a piece of software was accessed exactly 180 days ago, it should remain on the system.

Your task is to write a program that reads software names and access dates and decides which pieces of software to remove from a system based on the rules above.

**Program Input**

The file Prob05.in.txt will contain a reference date that the log files were collected on the first line, followed by a list of software names and access dates separated by a colon. There will be one name-date pair per line.

**Example Input:**

3/21/2013

WordZ:10/18/2012

CalculateThat:1/19/2010

DrawItWithThis:12/19/2012

Software 42:7/23/2012

**Program Output**

Your program’s output should display the software which is eligible to be removed and how many days it’s been since the software was accessed separated by a comma and a space. You should print one software-days pair per line, and your output should be in the same order as the input file.

**Example Output:**

CalculateThat, 1156

Software 42, 241

**Java program:** Prob06.java

**Input File:** Prob06.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

A golf tournament has just completed and everyone is eagerly waiting for the scores to be tabulated so they can eat the free chicken dinner. The crowd is getting restless and the waiters are starting to sweat. Write a program that can compute the scores from today’s round of golf based on the input file containing the golfers’ scores and report back the top three scores and the worst score.

**Program Input**

The file Prob06.in.txt will contain the list of golfers and their scores for the day. No two golfers will have the same name, and you can assume that there will be no ties for first, second, third, or last place. Each line will contain the golfer’s name followed by a colon and a comma delimited list of their scores. There are 18 holes, and each golfer will play all holes.

**Example Input:**

Bob Jones:4,6,3,3,4,3,5,5,4,4,3,3,2,3,4,3,4,4

Ted Smith:3,3,3,3,4,3,5,5,4,3,3,3,2,3,4,3,4,4

Taylor Martin:4,4,3,3,4,3,5,6,4,4,4,3,2,3,5,3,5,4

Adam Lee:4,4,3,3,4,3,5,5,4,3,3,3,2,3,4,3,4,4

Ben Gonzales:4,4,4,3,4,3,5,5,4,3,3,3,2,3,4,3,4,5

Brian Foster:6,5,5,4,4,4,5,5,4,4,4,3,3,3,4,3,5,4

Julian Perez:4,4,4,4,4,4,5,7,4,4,4,4,3,3,4,3,4,4

Mike Davis:3,4,4,4,4,3,5,5,4,3,3,3,2,3,4,3,4,4

Bob Taylor:3,4,3,3,4,3,5,5,4,4,4,4,2,3,4,3,4,4

Chris Matthews:3,3,4,4,4,4,5,5,4,4,4,3,4,3,5,4,5,5

**Program Output**

Your program should output the first, second, third, and last place golfers and scores by outputting the place in all capital letters followed immediately by a colon, and then followed immediately by the player’s name for that place. Remember, in golf the lowest score wins!

**Example Output:**

FIRST:Ted Smith

SECOND:Adam Lee

THIRD:Mike Davis

LAST:Brian Foster

**Java program:** Prob07.java

**Input File:** Prob07.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Brenda Sue just opened a new cupcake store. Brenda is more interested in baking than handling the money, and wants to make it easier for everyone to give the correct change. Write a program for Brenda Sue that determines the correct amount of change to return to her customers.

**Program Input**

The file Prob07.in.txt will containa list of amounts customers owe and how much money the customers gave Brenda Sue. There will be one pair of numbers per line, and the numbers will be separated by a comma and a space. Both numbers will always have two decimal places.

**Example Input:**

3.54, 10.00

16.87, 20.00

1.23, 5.00

4.78, 5.03

**Program Output**

Your program’s output should display the amount of money returned to the customer, and a list of the monetary values returned all separated by a comma and a space. An equal sign with no extra space should be between the denomination and the integer number corresponding to how many of that denomination to give to the customer. The change should always be broken down by largest possible denomination; the customer doesn’t want excess change or bills. For example, a quarter should be used instead of 5 nickels. The list of valid denominations is: TWENTY, TEN, FIVE, ONE, QUARTER, DIME, NICKEL, PENNY. If no change is owed, print the word NONE.

**Example Output:**

$6.46, FIVE=1, ONE=1, QUARTER=1, DIME=2, PENNY=1

$3.13, ONE=3, DIME=1, PENNY=3

$3.77, ONE=3, QUARTER=3, PENNY=2

$.25, QUARTER=1