WIC Beginner's Programming Competition

Spring 2014

Problem Set

DO NOT OPEN UNTIL TOLD TO DO SO

Problem 1: Eliminating Threes

Given an integer array, remove all multiples of 3, before '3' itself if it exists in the given array. That is that if the integer 3 exists in the array, remove all integers that are a multiple of 3 before it. Note that '0' is a multiple of all numbers.

Example(s):

```
int[] arr = [1, 2, 4, 6, 9, 8, 6, 3, 4, 9, 5]
return => [1, 2, 4, 8, 3, 4, 9, 5]

int[] arr = [3, 22, 6, 6, 9, 10, 4, 3, 5]
return => [3, 22, 6, 6, 9, 10, 4, 3, 5]
```

Sample File Format:

sample-in: One line per input. This line contains a series of integers separated by a space representing the input array.
sample-out: One line per output. This line contains a series of integers separated by commas, representing the returned array.

Required Method Signature:

```
// Parameter(s): list – array of integers that are to be altered
//                      and returned
// Return: int[] – the modified version of list
public static int[] eleminatingThrees(int[] list) {
     //TODO
}
```

Problem 2: Sand King Cometh

Sand King is in dire need of a theme song, and he stumbles upon a song called Sandstorm by Darude. After listening to the song, Sand King is determined that this will become his theme song. Nonetheless, as part of procedure, Sand King must determine the number of each lyrical word that is in the song. He hires you to do so.

You determine that Darude is composed of exactly three lyrical words: du, duu, duuu. Your job is to count the number of each in the lyrics. You are given the lyrics that have been formatted to contain no spaces. Return a string composed of three integer values separated by a single space. The first being the number of times "du" occurs, the second the number of times "duu" occurs, and the third representing the number of times "duuu" occurs.

Example(s):

```
String lyrics = "dududududuu"
return => "4 1 0"

String lyrics = "duuuduuuduuu"
return => "0 0 3"
```

Sample File Format:

sample-in: One line per input. This line contains a string containing the lyrics of the song, Darude.
sample-out: One line per input. This line contains a string representation of the number of repetitions of the lyrical words, du, duu, and duuu.

Required Method Signature:

```
// Parameter: lyrics - string containing the original lyrics
// Return: string - string containing the repetitions of the
//                  lyrics
public static String dududarude(String lyrics)
{
     // TODO
}
```

Problem 3: Letter Holes

The Chef at the Bistro wrote some text on a piece of paper and now she wants you to help her figure out how many holes are in the text. What is a hole? If you think of the paper as the plane and a letter as a curve on the plane, then each letter divides the plane into regions. For example the letter "A" divide the plane into two regions so we say it has one hole. Some letters may even have more than one hole, and others may have none, like "C", "E", "F", and "K". We say that the number of holes in the text is equal to the total number of holes in the letters of the text. Given an array of strings help the Chef to determine how many holes are in each element. Output an array with the number of holes for each string in the input array mapped to the corresponding index in the output array. It is given that the text only uses capital letters.

Example(s):

```
String[] list = ["DRINK","EAT","CODE"]
return => [2, 1, 2]  //DRINK is at index 0 and has 2 holes,
                     //therefore return an array with '2' at index 0.

String[] list = ["KEEP CALM AND CODE ON"]
return => [7]
```

Sample File Format:

sample-in: One line per input. A String array which contain the strings separated by commas.
sample-out: One line per input. An int array that stores the return values of each element given in the input.

Required Method Signature:

```
// Parameter(s): String[] list - words to find the number of
//                               holes in
// Return: int[] - holes in each word
public static int[] letterHoles( String[] list ) {
     //TODO
}
```

Problem 4: Counting Words

Given a sentence, count the numbers of words in the string. What defines a word? Words are a series of alphabetic characters that can be separated by spaces and/or punctuations, which include, but are not limited to: |,|.|\|/|?|!|:|;|. Alphabetic characters are the following: ‘a’-‘z’ and ‘A’-‘Z’.

Example:

```
“Hello, I am T900.” => 3
“fho3a ol etyy” => 2
```

Sample File Format:

sample-in: One line per input. This line contains a string from which the number of words needs to be counted.
sample-out: One line per input. This line contains how many words are in the given string.

Required Method Signature:

```
// Parameter(s): String sentence – from which the number of
//                                 words should be counted
// Return: int - number of words in the given sentence
public static int countWords(String sentence) {
     //TODO
}
```

Problem 5: Cooling Pies

The Chef has just finished baking several pies, and it's time to place them on cooling racks. The Chef has exactly as many cooling racks as pies. Each cooling rack can only hold one pie, and each pie may only be held by one cooling rack, but the Chef isn't confident that the cooling racks can support the weight of the pies. The Chef knows the weight of each pie, and has assigned each cooling rack a maximum weight limit. You will be given the number of pies, n, and two arrays. A n-length int array, that contains the weights of n pies, and a m-length int array that contains the weight limits for m cooling racks. Your job is to help the Chef figure out what is the maximum number of pies the chef can cool on the racks?

Example(s):

```
int n = 3
int[] pie_weights = [ 10, 30, 20 ]
int[] cooling_rack_weight_limit =  [ 30, 10, 20 ]
return => 3

int n = 5
int[] pie_weights = [ 9, 7, 16, 4, 8 ]
int[] cooling_rack_weight_limit =  [ 8, 3, 14, 10, 10 ]
return => 4
```

Sample File Format:

sample-in: Three lines. The first line contains an int N, where N <= 30, representing the number of pies. The second line contains the weights of each of each of the pie separated by a space. The third line contains the weight limits of each of the cooling racks can support separated by spaces.
sample-out: One line. This line contains an int answer representing the maximum number of pies the Chef can place on the rack with the given constraints.

Required Method Signature:

```
// Parameter(s): int n - the number of pies given
//               int[] weights - the weights of each pie
//               int[] limit - the weight limits for each rack
// Return: int - number of pies that can be placed on the racks
public static int coolingPies(int n, int[] weights, int[] limit)
{
     //TODO
}
```

Problem 6: Kindergarten Troubles

You are a kindergarten teacher, and your job is to assess the difficulty of math problems. You come up with a criterion to base the difficulty of a problem. You determine the number of times you must carry when adding two numbers is proportional to the problem's difficulty. Therefore, you are tasked with writing a program that counts the number of times you have to carry.

Example(s):

```
   999
+    1
--------
  1000
return => 3

   123
+    7
--------
   130
return => 1
```

Sample File Format:

sample-in: One line per input. This line contains two number separated by a single space, representing the numbers to add.
sample-out: One line per input. This line contains an integer representing the number of times you have to carry to add the given numbers.

Required Method Signature:

```
// Parameter: int n1 - first integer to add
//            int n2 - second integer to add
// Return: int – The number of times you must carry to add n1
//               with n2
public static int kindergarten(int n1, int n2)
{
     // TODO
}
```

Problem 7: Smart Vending Machine

Supposedly, there is a special-programmed vending machines located in the CSE building at UCSD. The vending machine is filled with 200 cans of Coca-Cola at the beginning of everyday. The starting price of each can is $1.50. Every time the machine sells 10 cans, it raises the price by 10 cents. So after the first ten cans are sold, the price is raised to $1.60, after 20 cans are sold the price is $1.70, and so-on. Given the numbers of cans sold each day of a week in an int array of length 7, your mission is to calculate how much money the vending machine earned for the given week.

Example:

```
[12, 13, 15, 21, 0, 6, 10] => 117.70
[1, 10, 4, 9, 20, 3, 9] => 85.00
```

Sample File Format:

sample-in: One line per input. This line contains seven numbers separated by spaces, indicating how many cans a vending machine sells in each day of the week.
sample-out: One line per input. This line contains a double representing the profit made by the end of the week.

Required Method Signature:

```
// Parameter(s): int[] cans – number of cans sold each day of a
//                            given week
// Return: double – profit made
public static double calculateProfit(int[] cans) {
     //TODO
}
```

Problem 8: Chef's Tree

The Chef is climbing a tree with infinite levels. At each level she has 3 branches she can choose to take, after picking one and increasing a level, she has 3 more that she can pick from and so on. Given an integer indicating the level chef is currently on, return the number of potential paths that she did not take.

Example(s):

```
int level = 2  => 8
int level = 1  => 2
```

Sample File Format:

sample-in: One line per input. This line contains a single integer representing the level chef is on that is to be used to find how many potential paths have not been taken.
sample-out: One line per output. This line contains an integer representing the number of potential paths that were not taken.

Required Method Signature:

```
// Parameter(s): int level – what level of the tree that chef is
//                           currently on
// Return: int – the number of paths chef did not take
public static int chefsTree(int level) {
     //TODO
}
```

Problem 9: Double Strings

The palindrome is a string that can be read the same way from left to right and from right to left. For example, strings "aaaaa", "1221", "bbaabb" are palindromes, however the string "chef" is not a palindrome because if we read it from right to left, we will obtain "fehc" that is not the same as "chef".

We call a string a "double string" if it has an even length and the first half of this string is equal to the second half of this string, for example "abab" is a double string because the first half "ab" is equal to the second half "ab", however the string "abba" is not a double string because the first half "ab" is not equal to the second half "ba". The empty string "" is a double string, and its length is 0.

The Chef doesn't like palindromes, however she likes "double strings". For this reason when she encounters a palindrome, she likes to change the order of letters and sometimes even remove some symbols from it to create "double strings". Given a palindrome String, help the Chef determine all of the possible "double strings" that can be made, by either switching the order of the characters and/or removing some characters. Return all of the possible "double strings" in a String array.

Example(s):

```
"abba"        => ["", "aa", "abab", "baba", "bb"]
"aaa"         => ["", "aa"]
```

Sample File Format:

sample-in: One line per input. This line contains a String representing a palindrome.
sample-out: One line per input. This line contains a list of Strings, representing the possible double strings derived from the palindrome separated by commas.

Required Method Signature:

```
// Parameter(s): String palindrome - given palindrome string
// Return: String[] - all of the possible double strings that
//                    can be derived from the input
public static String[] doubleStrings(String palindrome) {
     //TODO
}
```

Problem 10: When is Good?

Eight WIC's Beginner Programming Competition Committee members decide to meet some time today. Everyone has filled out a form about the time they are available to meet – '1' represents that they are available, and '0' represents when they are NOT available. Suppose there are 12 time slots provided on the form from 3:00 pm to 9:00 pm, each 30 minutes long. That is that for each row in the 2x2 array, the first element represents the time slot from 3:00 – 3:30, the second from 3:30 – 4:00, and so on until 9:00 p.m. Find out the start time and the end time of longest time period that all the members are available. If there is no time slot that everyone is available, return 0:00pm for both start time and end time.

Note, in the case that there are multiple optimal time slots, choose the time slot that is earliest in the day. For example if the longest time slot that everyone is free is for the duration of two hours, and occurs twice – once from 3:00 p.m. to 5:00pm, and the second from 6:00 p.m. to 8:00 p.m. – return the earlier time, "3:00pm-5:00pm".

Example(s):

```
int[][] info = {0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
                1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
                1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0}
String return = "6:30pm-7:30pm"

int[][] info = {0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
                1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
                1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
                1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0}
String return = "4:30pm-5:00pm"
```

Sample File Format:

sample-in: Eight lines per input. Each line has the availability information for one member.
sample-out: One line per output. The line contains the string showing the start and end time.

Required Method Signature:

```
// Parameter(s): int[][] info - the availability info 2-D array
// Return: String that shows the start time and end time public
static String whenIsGood(int[][] info) {
     //TODO
}
```

Problem 11: Big Fan of N

During her spare time, the Chef likes to play a game that is called big Fan of N. To play this game, the Chef needs 25 cards. Each card has a number on it. 5 cards have the number '1', 5 have '2', 5 have '3', 5 have '4', and the last 5 have '5'. Each round the Chef will randomly draw 5 cards, and use them to see if she can add them up to get the value N. Her goal is to see how many, if any at all, ways are there to add up to N with the chosen 5 cards. A card can be used in different groups but can only be used once in one set.

In this problem, given 5 random chosen cards in an int array, group the cards into independent sets that can add up to the given value for N. Return an integer array of the resulting groupings. Be sure to represent each number in your resulting array in ascending order – that is that if N = 10, and you have 3 cards as such: '3', '2', and '5', give the result as '235' where the cards are in ascending order.

Example(s):

```
int[] a = [1, 2, 3, 4, 5]
int n = 10
return => [235, 1234]

int[] a = [2, 2, 4, 5, 3]
int n = 8
return => [35, 224]

int[] a = [2, 1, 4, 2, 4]
int n = 8
return => [244, 244]
```

Sample File Format:

sample-in: Two lines per input. The first line has the int array separated by spaces. The second line has the number N.
sample-out: One line per output. This line contains the integer array that has all the combination of cards that add up to N separated by commas.

Required Method Signature:

```
// Parameter(s): int[] a – array that has the numbers on cards
//               int n – the number to add up to
// Return: int[] – array that has all the combination of cards
//                 arranged in ascending order.
public static int[] bigFanOfN(int[] a, int n) {
     //TODO
}
```

Problem 12: Divisor Sort

Given an array of integers, order them in terms of what they are divisible by (excluding 1) least to greatest. If a number is divisible by multiple numbers, choose the smallest one to determine its place. In the event of a tie, the smaller number is placed first.

Example(s):

```
[2, 13, 9, 8, 22]  => [2, 8, 22, 9, 13]
[2, 3, 4, 5, 6]    => [2, 4, 6, 3, 5]
```

Sample File Format:

sample-in: One line per input. This line contains a series of integers separated by spaces, representing the input array.
sample-out: One line per output. This line contains a series of integers representing the resulting array separated by commas.

Required Method Signature:

```
// Parameter(s): int[] list - list of ints that are to be sorted
// Return: int[] - the sorted ints from list
public static int[] divisorSort(int[] list) {
     //TODO
}
```

Problem 13: Footsteps

Foot steps. You're given an "Ascii picture" of footsteps. Your job is to calculate the number of footsteps in the image. A footstep is defined as a flood fill (up down left right) of the Ascii image represented in a 5x7 2D array. The char array will contain a '#' for each filled block, and '_' representing an empty block.

Example(s):

```
char[][] footsteps = { _, #, _, _, _, _, _
                       _, _, _, _, #, _, _
                       _, _, _, _, _, _, _
                       _, #, _, _, _, _, _
                       _, _, _, _, #, _, _ }
return => 4

char[][] footsteps = { _, _, #, #, #, _, _
                       _, #, #, #, #, #, _
                       _, _, #, #, #, _, _
                       _, _, #, #, #, _, _
                       _, _, _, #, _, _, _ }

return = 1
```

Sample File Format:

sample-in: Five lines per input. These lines contain the char array representing the "Ascii" image. Each line contains 7 symbols separated by a space, representing each row of the image.
sample-out: One line per input. This line contains an integer of number of footsteps.

Required Method Signature:

```
// Parameter: char [][] footsteps
// Return: int - number of footsteps
public static int footsteps(char [][] grid) {
     // TODO
}
```