ProblemSet[] = new ProblemSet[8]

Problem Set 8 – Arrays

This assignment must be completed and submitted before 11:59pm on Friday 12/23/2016.

**Part A: Reading and Review Questions (10 points total)**

Read Chapter 10.1-10.9 (pages 341 to 359) of Fundamentals of Java and answer the review questions below in a Microsoft Word document and save as:

**partA.yourFirstNameLastName.doc**

1. Can you change size of array once created? **No. Once an array is declared whatever size the number of values cannot be changed.**
2. How do you make a copy of an Array in Java? **copyOf() method copies the specified range of the specified array into a new array**
3. What is a for-each loop used for? In what situations will it work and under what conditions will it not work? **A for-each loop is used mainly used to traverse array one by one in order. A for-each won’t work where access is anything else since you are not specifically programming and controlling the index as you are with a for loop.**

**Part B: Simple Exercises (20 points total)**

Type up answers to the following nine problems in a simple “text file” without writing complete Java programs and saving as:

**partB.yourFirstNameLastName.txt**

Problem 1: (6 points)

Write Java variable declarations to create declarations for each array below. The arrays DO NOT need to be initialized.

a) a single-dimensioned array with the answers to 20 true/false quiz questions.   
**Boolean [] answers = new Boolean [20];**

b) a single-dimensioned array that contains the average family size in the years 1900, 1910, 1920, …, 2010  
**Double [] average = new Double [12];**

c) a single-dimensioned array containing the 8 candidates in some election. You should define a Candidate class with private instance variables for the person’s name and the number of votes received. Don’t bother defining any constructors, setters, getters or other instance methods.  
**//a single-dimensioned array containing the 8 candidates in some election.**

**Candidate [] election = new Candidate [8];   
public class Candidate {**

**//You should define a Candidate class with private instance variables for the person’s**

**//name and the number of votes received. Don’t bother defining any constructors,**

**//setters, getters or other instance methods.**

**private String name="";**

**private int numOfVotes=0;**

**}**

Problem 2: (4 points)

Suppose that “final” integer SIZE has been properly declared and initialized, and that an array named sample has been declared as follows:

int [] sample = new int [SIZE];

Write one or more Java statements to perform the following. Note that each task is independent of the other three:

a) Initialize all array elements to the value 5.  
 **for (int i = 0; i < sample.length; i++) {**

**sample [i] = i;**

**}**

b) “Swap” the first value in the array with the last value in the array.

**int tmp = sample [0];**

**sample [0] = sample [sample.length-1];**

**sample [sample.length-1] = tmp;**

c) Change any negative values to positive values (of the same magnitude).

**for (int i = 0 ; i < sample.length; i++) {**

**if (sample[i] < 0) {**

**sample[i] = -sample[i];**

**}**

**}**

d) Print the contents of the “odd-numbered” locations in the array.

**for (int index=1; index < sample.length; index=index+2) {**

**System.out.println(sample [index]);**

**}**

Problem 3: (6 points)

Write an enhanced-for loop (for-each loop) for each situation below.

1. A basketball player’s points scored in each game are stored in an int [] named points. How many points did he score in the season?

**int [] points = {1,3,5,7,9}; //5 game season**

**int total=0;**

**for (int p : points) {**

**total = total + p;**

**}**

1. A library has all book titles stored in a String[] named titles. Print each title on its own line

**String[] titles = {"Java Dummy","Dead Sea Scrolls","Holy Bible","Koran","Talmud"};**

**for (String t : titles) {**

**System.out.println(t);**

**}**

1. An dating website has a Person[] named singles. Instead of combing through the entire list, your older sister only wants to print the names of eligible bachelors who are between 25 and 29 years old. Assume the Person class has a getGender method which returns a String and getAge which returns an int.

**Person a = new Person ("Archie","m",25);**

**Person b = new Person ("Betty","f",29);**

**Person [] singles = {a,b};**

**for (Person p : singles) {**

**if (p.getGender() == "m"**

**&& p.getAge() >= 25**

**&& p.getAge() <= 29)**

**System.out.println(p.getName());**

**}**

Problem 4: (4 points)

The following simple program, which is supposed to create a 20-element array that it fills with random double values, contains a number of distinct errors. Some of the mistakes will cause an error message at compilation time, while others may cause an error at execution time. Find (and fix) all of them!

public class Foobar

{

public static void main (String [] args)

{

**double [] r = new double [20];** ~~//double [20] r;~~

for (int i = 0; i <= r.length**~~/\*()\*/~~**; i++) **{**

**//length is a field not a method**

r[i] = Math.random**()** \* 100;

**}**

}

}

**Part C: Programming Problems (70 points total)**

Problem 1: (10 points) Use file **RepeatedDigits.java**

Write a Java program that allows the user to input an integer value; after this number is entered, your program will print out how many times each of the digits from 0 through 9 appeared in your input value. Here is how your program might appear in action:

Please enter an integer: **3456353**

Digit: 0 1 2 3 4 5 6 7 8 9

Occurrences: 0 0 0 3 1 2 1 0 0 0

Note:

* leading zeros should be ignored (000023 is the same as 23)
* negative numbers can be considered valid input by somehow ignoring the “-” sign.

The occurrences should be stored in a single array.

Problem 2: (10 points) Use file **ArrayIntersect.java**

Write a Java method named **intersection** that accepts two arguments, both arrays of integers; these arrays may contain a different number of elements from each other.

Your method must return the value **true** if there exists an integer value, k, that is contained in both tables. If no such integer exists, then your method should return the value **false**. You are not to assume that k is any integer value in particular! Write a **main()** method which shows off the features of your method.

Problem 3: (15 points) Use file **TestStretch.java**

Write a static method named stretch that accepts an array of integers as a parameter and returns a new array that is twice as large as the original, replacing every integer from the original array with a pair of integers, each half the original.

If a number in the original array is odd, then the first number in the new pair should be one higher than the second so that the sum equals the original number.

For example, if a variable named list refers to an array storing the values {18, 7, 4, 24, 11}, the call of stretch(list) should return a new array containing {9, 9, 4, 3, 2, 2, 12, 12, 6, 5}. (The number 18 is stretched into the pair 9, 9, the number 7 is stretched into 4, 3, the number 4 is stretched into 2, 2, the number 24 is stretched into 12, 12 and the number 11 is stretched into 6, 5.)

Test your code with the following class:

import java.util.\*;

public class TestStretch

{

public static void main(String[] args)

{

int[] list = {18, 7, 4, 14, 11};

int[] list2 = stretch(list);

System.out.println (Arrays.toString(list));

// the above prints [18, 7, 4, 24, 11]

System.out.println (Arrays.toString(list2));

// the above prints [9, 9, 4, 3, 2, 2, 12, 12, 6, 5]

}

**// your code goes here**

}

Problem 4: (15 points) Use file **MindTheGap.java**

Write a static method named **minGap** that takes an integer array as a parameter and that returns the *minimum gap between adjacent values in the array*. The gap between two adjacent values in a list is defined as the second value minus the first value. For example, suppose a variable called "list" is an array of integers that stores the following sequence of values:

{1, 3, 6, 7, 12}

The first gap is **2** (3 - 1), the second gap is **3** (6 - 3), the third gap is **1** (7 - 6) and the fourth gap is **5** (12 - 7). Thus, the call **minGap (list)** should return 1 because that is the smallest gap in the list. Notice that the minimum gap could be a negative number. For example, if list stores the following sequence of values {3, 5, 11, 4, 8} then the gaps would be computed as 2 (5 - 3), 6 (11 - 5), -7 (4 - 11), and 4 (8 - 4). Of these values, -7 is the smallest, so it would be returned.

By the way, this gap information can be helpful for determining other properties of the list. For example, if the minimum gap is greater than or equal to 0, then you know the list is in sorted (nondecreasing) order. (Do you understand why?) If the gap is greater than 0, then you know the list is both sorted and contains unique values (i.e., it is strictly increasing).

If your method is passed an array with fewer than 2 elements, minGap should return 0. Be sure to convincingly test your method out on a variety of arrays using a main program of your own design.

Problem 5: (20 points) Use file **MyersBriggs.java**

Write a program to score users’ responses to the classic Myers-Briggs personality test. Assume that the test has 70 questions that determine a person’s personality in four dimensions. Each question has two answer choices that we’ll call the “A” and “B” answers. Questions are organized into 10 groups of seven questions, with the following repeating pattern in each group:

* The first question in each group (questions 1, 8, 15, 22, etc.) tells whether the person is extroverted or introverted.
* The next two questions (questions 2 and 3, 9 and 10, 16 and 17, 23 and 24, etc.) test whether the person is guided by his senses or his intuitions.
* The next two questions (questions 4 and 5, 11 and 12, 18 and 19, 25 and 26, etc.) test whether the person focuses on thinking or feeling.
* The final two questions in each group (questions 6 and 7, 13 and 14, 20 and 21, 27 and 28, etc.) test whether the person prefers to judge or be guided by perception.

In other words, if we consider extraversion/introversion (E/I) to be dimension 1, sensing/intuition (S/N) to be dimension 2, thinking/feeling (T/F) to be dimension 3, and judging/perception (J/P) to be dimension 4, the map of questions to their respective dimensions would look like this:

1223344122334412233441223344122334412233441223344122334412233441223344

BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA

The following is a partial sample input file of names and responses:

Chewbacca

BABAAAABAAAAAAABAAAABBAAAAAABAAAABABAABAAABABABAABAAAAAABAAAAAABAAAAAA

Han Solo

AABBAABBBBBABABAAAAABABBAABBAAAABBBAAABAABAABABAAAABAABBBBAAABBAABABBB

If the person has less than 50% B responses for a given personality dimension, her type for that dimension should be the first of its two choices. If the person has 50% or more B responses, her type for that dimension is the second choice. Your program should output each person’s name, number of A and B responses for each dimension, percentage of Bs in each dimension, and overall personality type. The following should be your program’s output for the preceding input data:

Chewbacca:

1A-9B 17A-3B 18A-2B 18A-2B

[90%, 15%, 10%, 10%] = ISTJ

Han Solo:

7A-3B 11A-9B 14A-6B 6A-14B

[30%, 45%, 30%, 70%] = ESTP

If you would like to learn more about your personality, you can try a variant of this test at:

<http://www.16personalities.com/free-personality-test>

Problem 6: EXTRA CREDIT: 10 points Use file **Bool.java**

As a practical joke for your birthday, your best friend gave you some single dimensioned arrays, each full of boolean values. She didn’t tell you how many items are stored in these arrays.

As if that weren't bad enough, your friend told you that each array is filled almost entirely with false values, except for a small sequence of consecutive trues contained somewhere in the array.

Overwhelmed by the novelty of all this, you decided to write a static Java method named fullOfBool that accepts an array of Boolean values as its only argument, and will print out

* the location of the first true in the boolean array
* the location of the last true in the boolean array
* the total number of true values in the string of consecutive true values.

Your main method should call on fullOfBool using the following arrays to test it out:

boolean [] test1 = { false, true, true, true } ;

boolean [] test2 = { true } ;

boolean [] test3 = { true, true, true, true, false } ;

**

*You are finished! Create a “.zip” of your work so that you can electronically submit it on Schoology. Click the link that says “Problem Set 8 Dropbox.” and attach the zip file.*