**CSCI 1100 – 2017**

**Laboratory Report 9**

**Name:**

**Student ID:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Please indicate your registered lab room number:** | | | |
| **Rm 133** | **Rm 134** | **Rm 143** |  |

|  |  |  |
| --- | --- | --- |
| **Declaration**  **Complete this section by filling in the shaded column to accurately reflect your work** | | |
| 1 | This document is entirely my own work. | *Yes/no* |
| 2 | I obtained some help to complete this document. | *Yes/no.*  *If yes, from whom? Give details.*  *It is reasonable to obtain help from any person as long as you acknowledge the source.* |
| 3 | This document contains some material from the Internet or another document, file, or program. | *Yes/no.*  *If you got help from a source you need acknowledge this (e.g., the URL or a book reference). You must detail how you used this source.*  *Your lab work must still be your own. You should* ***not*** *copy and paste partial or full solutions.* |

Your task is to complete this report using Word and JGrasp. You may use your own computer, or one of the lab computers provided.

Your submission should be a **ZIP** file containing your source code files. You should submit your **ZIP file** on Brightspace:

[http://dal.brightspace.com](http://dal.brightspace.com/)

**Submission Deadlines (firm):**

Monday Labs due: Wednesday by 12:00pm (**noon**)

Friday Labs due: Sunday by 12:00pm (**noon**)

NB:

* Try to submit this report *during* the lab so that your TA can check it for you before you submit!
* Attendance is mandatory in all labs, and will form part of your overall lab grade
* Acknowledge any help that you obtained from friends, TAs, the Learning Centre, etc., by completing the Declaration on the first page of this document. Obtaining help is fine, *so long as you acknowledge it!*
* Any students who cannot log on to the lab computers should speak to the Help Desk to set up their account.
* Textbooks, class handouts, and any other materials are welcomed and encouraged in all labs!
* Food and drink are not permitted in the computer labs
* Late labs are ***not*** accepted! It is known that computer errors, power outages, and network lag are 105% more likely to occur between 11:55-11:59am, in the moment they can do the most damage. Account for this, and give yourself the chance to make a timely submission!

**Exercise 1.**

1. Write a method mysteryApprox that takes one integer parameter and returns a double value. The method should compute a value that is the sum of alternating fractions: 1, -1/3, 1/5, -1/7, 1/9, -1/11, etc., and return 4 times this sum. The number of alternating fractions to use is given by the integer parameter, and you may assume that it will be a non-negative value. For example, if called with the parameter 2, your method should return 4 times the sum of the first two fractions in the above pattern, which is 2.6666666666666667.
2. Experiment on your own (by calling mysteryApprox from the main method) and determine for yourself: what value does this method appear to approximate? Specify your hypothesized value in your code using a multi-line header comment for your method mysteryApprox.
3. Write a method called tripleProduct that takes a single integer parameter and returns an integer. The method should compute and return the integer value that is the product of the three consecutive integers starting with twice the parameter value. For example, if called with the parameter 1, your method should return the product of 2, 3, and 4 (which is 24), and if called with the parameter 3, your method should return the product of 6, 7, and 8 (which is 336).
4. Write a method called alternateSign that takes two integer parameters and returns an integer. The method should return a multiple of the second integer parameter, where the multiple will either be (+1) or (-1). Your method should consider the first integer parameter, and use a multiple of (+1) if the first integer parameter value is odd, or use a multiple of (-1) if the first integer parameter value is even. For example, if called with the parameters (3, 5), your method should return 5, and if called with the parameters (10, 20), your method should return -20.
5. Write a method called makeFraction that takes a single integer parameter and returns a double. The method should compute and return the multiplicative inverse of the integer parameter value. For example, if called with the parameter 5, your method should return 0.2, and if called with 100, your method should return 0.01.
6. Write a method called secondMysteryApprox that takes one integer parameter and returns a double. Your method should compute the value that is the sum of values given by repeated calls to the method given below: computeStep(0), computeStep(1), computeStep(2), etc. The number of calls to the method is given by the integer parameter, and you may assume that it will be a non-negative value. (You should copy-and-paste or type exactly the following method into your code):

public static double computeStep(int step) {

if(step == 0)

return 3.0;

else

return 4.0 \* makeFraction(alternateSign(step, tripleProduct(step)));

}

1. Use the main method to call your method secondMysteryApprox with parameter 100 and then with parameter 500, and observe the returned values. What value does this method appear to approximate? Specify in your code using a multi-line header comment for your method secondMysteryApprox.

For this question and all following questions:

* remember to use comments in your code where appropriate!
* include your source code in a class named after the exercise number (e.g., Exercise1)

**Exercise 2.**

1. Write a method called checkPrice that takes a single String parameter and returns a double. The method should determine whether the passed String is in a local variable array of Strings, and if so, will return a corresponding price. Your method should store prices for several different kinds of fruit: Apples, Oranges, Bananas, and Melons, which cost $0.25, $0.50, $0.75, and $0.99 each, respectively. If the passed parameter String matches any of these fruit names (ignoring case), then the corresponding price should be returned; otherwise, a price of $0 should be returned.
2. Write a method called calculateTotal that takes an array of Strings and an array of integers as parameters and returns a double. The method should calculate the total price of an order of fruit, using the parameter array of Strings to determine which kinds of fruit are being ordered, and the parameter array of integers to determine the quantities desired. Use calls to the method from part (a) to calculate the total price of the order. The parameter arrays may be of any legal size, and have no restrictions on their possible values (so for instance, “Clementines” may be one of the elements in the parameter array of Strings). Use a reasonable approach for determining which integer quantities correspond to which String fruit name.
3. In the main method, create two different arrays of Strings containing fruit names and two different arrays of integers containing (non-negative) quantities. Your arrays of Strings should each contain at least one String that is not Apples, Oranges, Bananas, or Melons. Use these arrays (once each) to make two calls to the method form part (b).

**Exercise 3.**

1. Write a method that collects an integer value as input using the Scanner class. Use either a while loop or a do-while loop to continue collecting values from the user until the value is a multiple of 11. At each iteration, give a meaningful prompt to the user as output. Return this value.
2. Write a method that collects a String as input using the Scanner class. Use either a while loop or a do-while loop to continue collecting values from the user until the given String has more than 20 characters. At each iteration, give a meaningful prompt to the user as output. Return this value.
3. Call each method once from the main method.

**Exercise 4.**

Write a program that will be used for booking seats on a small local airline. The seats are numbered from 1-20, and are spread across 5 rows (4 seats per row). Read the following method outlines, then decide on a representation of the airplane seat plan, and keep track of this throughout your main method. You will make multiple calls to several of these methods of your own design, simulating the process of booking seats on the airplane. At the end, you will be able to print out which seats are booked or available in which row, and by what kind of customer.

1. Write a method that checks whether a specific seat number is reserved or available.
2. Write a method that books a seat for a passenger, if it is available, and records whether the passenger is an adult, senior, or child. Use any representation you wish for distinguishing between these three cases.
3. Write a method that counts the number of available seats in a given row (row numbers will be between 1 and 5).
4. Write a method that prints off all the seats in a given row: this should indicate whether each seat is unoccupied, reserved for an adult, reserved for a senior, or reserved for a child.
5. Write a method that books a seat for a passenger, if it is available, and records whether the passenger is an adult, senior, or child. This time, your method should expect that the seat will be specified by its row number and position within the row (***not*** by its actual seat number!). For instance, if an attempt was made to reserve the 4th seat in row 2, this would attempt to reserve seat number 8. You may make this reservation attempt directly within this method, or you may convert to the seat number and then use your method from part (b).

Before submitting, check that:

1. You have properly filled in your name, ID, and Declaration on the first page,
2. You have included your solution for each question that requires one,
3. Your solutions are easy to read and formatted appropriately,
4. You recited **π** from memory to the 20th decimal for your TAs (optional),
5. You have saved your submission (which should be a completed copy of **this file**) as a **PDF**,
6. You have included *all* of your source files (one per question) in a **ZIP** file,
7. You are preparing to submit **both** your **PDF** and **ZIP** file on Brightspace, and
8. You have logged off of any lab computers.