

## **Part I**

**Please read carefully:** You must submit the answers to all the questions below. However, this part will not be marked. Nonetheless, failing to submit this part fully will result in you missing 50% of the total mark of the assignment.

### **Question 1**

- a) Given an array of integers of any size,  $n \geq 1$ , write an algorithm as a pseudo code (not a program!) that would reverse every two consecutive elements of the left half of the array (i.e. reverse elements at index 0 & 1, at index 2 & 3, etc.). Additionally, for the right half of the array, the algorithm must change the second element of every two consecutive elements to have to the sum of the two elements. For instance, if the right half starts at index 14, and the values at index 14 & 15 are 72 and 11, then the value at index 15 is changed to 83. If the given array is of an odd size, then the exact middle element should not be manipulated by the algorithm in any way. Finally, your algorithm **must** use the smallest auxiliary/additional storage to perform what is needed.
- b) What is the time complexity of your algorithm, in terms of Big-O?
- c) What is the space complexity of your algorithm, in terms of Big-O?

### **Question 2**

Given a string of random length and random contents of characters, that do not include digits (0-9), write an algorithm, **using pseudo code** that will shorten the representation of that string by adding the number of consecutive characters. For instance, given *str* as “gggN@@@@@KKeeeejjdsmmu” the algorithm should return the following representation of the string: “g3N@5K2e4j2dsm2u”.

- a) What is the time complexity of your algorithm, in terms of Big-O?
- b) What is the space complexity of your algorithm, in terms of Big-O?

### **Question 3**

i) Develop a well-documented pseudo code that finds the two consecutive elements in the array with the smallest difference in between, and the two consecutive elements with the biggest difference in between. The code must display the values and the indices of these elements. For instance, given the following array [20, 52, 400, 3, 30, 70, 72, 47, 28, 38, 41, 53, 20] your code should find and display something similar to the following (notice that this is just an example. Your solution must not refer to this particular example.):

The two conductive indices with smallest difference between their values are: index 5 and index 6, storing values 70 and 73.

The two conductive indices with largest difference between their values are: index 2 and index 3, storing values 400 and 3.

In case of multiple occurrences of the smallest or largest differences, the code should display the first found occurrence.

ii) Briefly justify the motive(s) behind your design.

iii) What is the time complexity of your solution? You must specify such complexity using the Big-O notation. Explain clearly how you obtained such complexity.

iv) What is the maximum size of stack growth of your algorithm? Explain clearly.

## **Part II**

**Purpose:** Practically, this part of the assignment should be considered as *Programming Assignment # 0!* The purpose of this assignment is to help you review some of the main topics covered in previous courses, including classes, loops, arrays, arrays of objects, static attributes and static methods.

### **General Guidelines When Writing Programs:**

- Include the following comments at the top of your source codes  
// -----  
// Assignment (include number)  
// © Your Name  
// Written by: (include your name and student id)  
// -----
- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

### **Part II.A**

A **Computer** object has four attributes, a brand (String), a model (String), an SN (long), and a price (double). SN indicates the serial number of the computer.

For this part, you are required to design and implement the **Computer** class according to the following specifications:

- Upon the creation of a computer object, the object must immediately be initialized with valid values; that is brand, model, SN and price. (Hint: Constructors.)
- The design should allow enough flexibility so that the value of any of these attributes can be modified later on. For example, it should be possible to create a computer object with

a given price then change its price later on. The design should also allow the user to obtain the value of any of the attributes. (Hint: Accessors & Mutators.)

- The design should allow all information of an object to be displayed at once, through the utilization of **System.out.print()** method. (Hint: **toString()** method)
- It is required to know how many Computer objects have been created. For that, you need to add a method, called **findNumberOfCreatedComputers()**, to the class. This method must return the number of created Computer objects prior to the time this method is called. The method would simply return 0 if no computers have been created by the time the method is called. (Hint: You are allowed to add other attributes to the class.)
- It is required to compare two Computer objects for equality. Two Computer objects are considered equal if they have the same brand, model and price. (Hint: **equals()** method)

## **Part II.B**

You are the owner of a computerstore and need help in keeping track of your computers. Write a driver program that will contain the following methods at least. **Note:** You can have the main function in a separate driver file, or in the same file if you prefer.

### 1. a **main method**, that will:

- a. Display a welcome message
- b. Prompt the computerstore owner for the maximum number of computers (maxComputers) his/her computerstore can contain. Create an empty array, called **inventory**, that will have the potential of keeping track of the created Computer objects.
- c. Display a main menu (Fig 1) with the following choices, and keep prompting the user until they enter a number between 1 and 5 inclusive:

What do you want to do?

1. Enter new computers (password required)
2. Change information of a computer (password required)
3. Display all computers by a specific brand
4. Display all computers under a certain a price.
5. Quit

Please enter your choice >

Fig 1. Main menu

- d. When option 1 is entered:
  - i. Prompt the computerstore owner for his/her password. (Make sure you have a constant variable containing the password “**password**” – do not use any other password as it will be easier for the marker to check your assignments). The computerstore owner has 3 tries to enter the correct password. After the 3<sup>rd</sup> illegal entry, the main menu in figure 1 is re-displayed again.
  - ii. If the correct password is entered, ask the owner how many computers he/she wants to enter. Check to make sure that there is enough space in the computerstore

(array of **Computer**) to add these many computers. If so, add them; otherwise inform the owner that he/she can only add the number of remaining places in the array. (How the computer information will be input/entered by the user, is up to you).

e. When option 2 is entered :

- i. Prompt the computerstore owner for his/her password. (Make sure you have a constant containing the password “password” as a constant – do not use any other password as it will be easier for the marker to check your assignments). Again the computerstore owner has 3 tries to enter the correct password. After the 3<sup>rd</sup> illegal entry, the main menu in figure 1 is re-displayed again.
- ii. Ask the user which computer number he/she000 wishes to update. The **computer** number is the index in the array **inventory**. If there is no **Computer** object at the specified index location, display a message asking the user if he/she wishes to enter another computer, or quit this operation and go back to the main menu. If the entered index has a valid computer, display the current information of that computer in the following format:

Computer # X  
Brand: brand of this computer  
Model: model of computer  
SN: serial number (SN) of this computer  
Price: \$price

Then ask the user which attribute they wish to change by displaying the following menu.

What information would you like to change?

1. brand
2. model
3. SN
4. price
5. Quit

Enter your choice >

Fig 2. Update menu

Once the user has entered a correct choice, make the changes to the attribute then display again all of the attributes on the screen to show that the attribute has been changed. Keep prompting the user for additional changes until the user enters 5. Each time the user is prompted for a choice make sure that a number from 1 to 5 is entered, otherwise keep prompting until a valid number is entered.

- f. When option 3 is entered, prompt the user to enter a brand name. You then need to display the information of all computers that have that brand. (Hint: You may use a static method, for instance called **findComputersBy** , which accepts a string for a brand name then performs the needed search).

- g. When **option 4** is entered, prompt the user to enter a value (representing a price). You then need to display all computers that have a value smaller than that entered value. (Hint: You may use a static method, for instance called **findCheaperThan**, which accepts a double value, for a price, then performs the needed search).
- h. When **option 5** is entered, display a closing message and end the driver.

<b>Submitting Assignment 1</b>
--------------------------------

**You need to submit Part I and Part II separately.**

**Part I:**

- **Part I must be submitted individually (No groups are allowed) under the submission folder: Assignment 1 – Part I.**

**Part II:**

- **For Part II, a group of 2 (maximum) is allowed. No additional marks are given for working alone.**
- **If working alone, you need to zip (see below) and submit your zipped file under the submission folder: Assignment 1 - Part II.**
- **If working in a group, only one submission is to be made by either of the two members (do not submit twice). You need to zip (see below) and submit your zipped file, under the submission folder: Assignment 1 - Part II.**

**Submission format:** All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must also contain your name(s) and student ID(s). Use your “official” name only - no abbreviations or nick names; capitalize the usual “last” name. Inappropriate submissions will be heavily penalized. If working in a group, the file name must include both IDs and last names.

**IMPORTANT:** For Part II of the assignment, a demo for about 5 to 10 minutes will take place with the marker. You (or **both** members, if groups are permitted) **must** attend the demo and be able to explain their program to the marker. Different marks may be assigned to teammates based on this demo. The schedule of the demos will be determined and announced by the markers, and students must reserve a time slot for the demo (only one time slot per group). **Now, please read very carefully:**

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at the second appointment will result in zero marks and no more chances will be given under any conditions.**

<b>Evaluation Criteria for Part II (10 points)</b>
--

**IMPORTANT:** **Part I** must fully be submitted. Failure to submit that part will cost 50% of the total marks of the assignment!

<b>Part II.A (Class Computer)</b>	<b>4 pts</b>
Default & copy constructors	1 pt
Accessor/mutator method for static attribute	1 pt
equals, toString and static attributes/methods	2 pts
<b>Part II.B (Driver &amp; other static methods)</b>	<b>6 pts</b>
Handling of password	1 pt
Handling of option 1	1 pt
Handling of option 2	1 pt
Handling of option 3	1 pt
Handling of option 4	1 pt
Handling of option 5	1 pt