Dept. of Computer Science & Software Eng., Concordia University
COMP 6481 --- Winter 2021

Programming and Problem Solving

Assignment 1 --- Due Sunday, February 28, 2021

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 swap the first element with the last, second with (n-1)th, third with (n-2)th, and so on. Additionally, if the array is odd, the algorithm must replace the middle element with some of the first and last element. If the sum of these numbers is more than 9, the algorithm must add all the individual digits until you get a single digit sum. For instance, given [1, 3, 5, 7, 2, 4, 6], the algorithm will return [6, 4, 2, 7, 5, 3, 1] Finally, your algorithm **must** use the smallest auxiliary/additional storage.

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 special characters, write an algorithm, **using pseudo code** that will swap every alternate pair of consecutive characters, starting from the left. Additionally, every vowel must be replaced with the sum of digits of its ASCII value. For instance, given "assignment1" the algorithm should return the string: "s7s6ngm2tn1".

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 of their squares reversed, and the two consecutive elements with the biggest difference of their cubes reversed. The code must display the values and the indices of these elements. For instance, given the following array [20, 23, 21, 3, 50, 77, 72, 11, 29, 13] your code should find and display something like the following (notice that this is just an example. Your solution must not refer to this example.)

The two elements with smallest difference of their squares reversed are: "11" at index 7 and "29" at index 8. The two elements with the biggest difference of their cubes reversed are: "50" at index 4 and "77" at index 5. 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 can 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

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

✓ A book object has four attributes, a book *title* (String), an author *name* (String), an *ISBN* number (long), and a *price* tag (double).

- Upon the creation of a book object, the object must immediately be initialized with valid values; that is title, author name, ISBN number and price value. (Hint: use constructors.).
- The design should allow enough flexibility so that the value of any of these attributes can be modified later. For example, it should be possible to create a book object with a given price then change its price later. The design should also allow the user to obtain the value of any of the attributes. (Hint: use 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: use toString() method).
- It is required to know how many Book objects have been created. For that, you need to add a method, called findNumberOfCreatedBooks(), to the class. This method must return the number of created Book objects prior to the time this method is called. The method would simply return 0 if no books has been created by the time the method is called. (Hint: use Static – You can add other attributes to the class.).
- It is required to compare two Book objects for equality. Two Book objects are considered equal if they have the same ISBN and price. (Hint: use equals() method).
- It is required to display any Book object (all info of that object) using the System.out.println() method. (Hint: use toSting() method).

Part II B

You are hired by a bookstore to write a software application that helps the store staff (users) in keeping track of the books at the store.

Write a driver program that will contain the **main()**method and it will perform the following: (Note: You can have the main function in a separate driver file, or in the same file)

- Display a welcome message.
- Prompt the user for the maximum number of books (maxBooks) his/her bookstore can contain. Create an empty array, called inventory, that will have the potential of keeping track of the created Book objects.
- Display a main menu (figure 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 books (password required)
    2.  Change information of a book (password required)
    3.  Display all books by a specific author
    4.  Display all books under a certain a price.
    5.  Quit
Please enter your choice >
```

Figure 1. Main menu

- When option 1 is entered:
  - Prompt the user 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 user has a maximum of 3 attempts to enter the correct password. After the 3rd wrong attempt entry, the main menu in figure 1 is re- displayed again. Additionally, after this process is repeated 4 times (*i.e.*, total failed attempts are 12), the program must display following messages:

    "Program detected suspicious activities and will terminate immediately!", then the program must exits.
  - If the correct password is entered, ask the user how many books he/she wants to enter. Check to make sure that there is enough space in the bookstore (array of Book) to add that many books. If so, add them; otherwise inform the user that he/she can only add the number of remaining places in the array. (How the book information will be input/entered by the user, is up to you).
- When option 2 is entered:
  - Prompt the user for his/her password. (make sure you have a constant containing the password "password" as a constant – do not use another password). Again, the user has 3 attempts to enter the correct password. However, after the 3rd wrong attempt entry, the main menu in figure 1 is simply re-displayed again (notice the different behaviour in that case from the previous one above).
  - Once the correct password was entered, the user is asked which book number he/she wishes to update. The book number is the index in the array inventory.

If there is no Book object at the specified index location, display a message asking the user if he/she wishes to re-enter another book, or quit this operation and go back to the main menu. If the entered index has a valid book, display the current information of that book in the following format:

```
Book: # x     (index of the book in the inventory array)
Author: name of author
Title: title of book
ISBN: ISBN #
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.  author
    2.  title
    3.  ISBN
    4.  price
    5.  Quit
Enter your choice >
```

Figure 2. Update menu

- Once the user has entered a correct choice, make the changes to the attribute then display again all 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. (ensure that the user can change any of the choice 1 to 4 on figure 2).

➢ When option 3 (in the main menu shown in figure. 1) is entered, prompt the user to enter an author name. You then need to display the information of all books by that requested author. (Hint: You may use a static method, for instance called findBooksBy, which accepts a string for an author name then performs the needed search).

➢ When option 4 (in the main menu shown in figure. 1) is entered, prompt the user to enter a value (representing a price). You then need to display all books 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).

➢ When option 5 (in the main menu shown in figure. 1) is entered, display a closing message, and end the driver.

SUBMISSION INSTRUCTIONS

Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and Assignment name (i.e. A1) 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 **must** attend the demo and be able to explain their program to the marker. 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.**

EVALUATION CRITERIA

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

Part II.A (Class Computer)	**4 pts**
Default & copy constructors	1 pt
Accessor/mutator method for static attribute	1 pt
equals, toString and static attributes/methods	2 pts
Part II.B (Driver & other static methods)	**6 pts**
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