Department of Computer Science & Software Eng., Concordia University
COMP 6481 --- Fall 2023

Programming and Problem Solving

Assignment 1 --- Due Sunday, October 15, 2023

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 a list of integers of any size (n ≥ 1) and two numbers, A and B, develop an algorithm as **pseudocode** (not a program!) that swaps all occurrences of A with B and all occurrences of B with A in the list. You must perform this operation in place, without creating an additional array, and keeping the number of operations as small as possible. For example, given the list [5, 4, 6, 9, 6, 3, 6] and numbers A=6 and B=9, the algorithm will return [5, 4, 9, 6, 9, 3, 9]. Finally, algorithm must not use any 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 alphanumeric characters, write an algorithm in **pseudo code** that will rearrange the string such that all its lowercase letters come first, followed by all its digits in ascending order, and then all its uppercase letters in alphabetical order. For instance, given the input string "a4Zb2C", algorithm must return "abc24Z".

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) Design a **well-documented pseudocode** algorithm that finds two consecutive prime numbers in an array with their sum equal to a given target value. The algorithm should display the values and indices of these elements. For instance, given the following array [2, 7, 5, 11, 3, 17, 13, 19] and a target value of 12, your code should find and display something like the following (this is just an example; your solution must not refer to this example):

"Two consecutive prime numbers with a sum of 12 are 5 and 7, found at indices 2 and 1 respectively."

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 Cricketer class according to the following specifications:

- ✓ A Cricketer object has seven attributes, namely, a cricketerID (long), cricketerName (String), battingAvg (float), bowlingAvg (float), strikeRate (float), economyRate (float), and isAvaialable (boolean).
  - ➢ Upon creation of a Cricketer object, the object must immediately be initialized with valid default values for all the attributes. (Hint: use constructors.).
  - ➢ The design must allow enough flexibility so that value of any of these attributes can be modified later except the CricketerID. For example, one should be able to create a Cricketer object with a given battingAvg and then change it later. The design must allow a user to obtain value of any attributes. (Hint: use accessors & mutators.)
  - ➢ The design must also allow all information of an object to be displayed at once through the System.out.print() method. (Hint: use toString() method).
  - ➢ It is required to know how many Cricketer objects have been created so far at any time. For that, you must have a method called getNumberOfCricketers(), in the Cricketer class. This method must return number of Cricketer objects prior to the time this method is invoked. The method would return 0 if no Cricketers have 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 Cricketer objects for equality. Two Cricketer objects are considered equal if they have the same CricketerID as well as isAvaialable. (Hint: use equals() method).

Part II B

You are hired by a Cricket Organization to write a software application that helps their staff (users) in keeping track of the Cricketers registered with them.

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

- Display a welcome message.
- Prompt a user for maximum number of Cricketers (maxCricketers) an organization can manage. Create an empty array, called CricketerDatabase, that will have the potential of keeping track of all the created Cricketer 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 Cricketers (password required).
    2. Change information of a Cricketer (password required).
    3. Display available Cricketers with a bowlingAvg greater than user value.
    4. Display all Cricketers that can play as an AllRounder.
    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. Additionally, after this is repeated 4 times (*i.e.*, after 12 consecutive failed attempts), program must display following message:

    "Program detected suspicious activities and will terminate immediately!", then the program must exit.
  - If correct password is entered, ask the user how many Cricketers he/she wants to enter. Check to make sure that there is ample space in the CricketerDatabase (array of Cricketers) to add that many Cricketers. If yes, add them; otherwise inform the user that he/she can only add the number of remaining places in the array. (How the Cricketer information will be entered by the user, is up to you).
- When option 2 is entered:
  - Prompt the user for a 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, the main menu in figure 1 is simply re-displayed. (Notice the different behaviour in that case from the previous one above).
  - Once the correct password is entered, the user is asked which Cricketer he/she wishes to update. Cricketer will be identified by a CricketerID. If there is no Cricketer object with the specified CricketerID, display a message asking the user if he/she wishes to re-enter another Cricketer, or quit this operation and go back to the main menu. If the entered index has a valid Cricketer, display the current information of that Cricketer in the following format:

Cricketer: # x (index of the Cricketer in the CricketerBase array)
ID: CricketerID of the Cricketer
Name: CricketerName of the Cricketer
Batting Average: battingAvg of the Cricketer
Balling Average: bowlingAvg of the Cricketer
Strike Rate: strikeRate of the Cricketer
Economy Rate: economyRate of the Cricketer
Availability: isAvaialable

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

What information would you like to change?
1. Name
2. Batting Average
3. Bowling Average
4. Strike Rate
5. Economy
6. Availability
7. 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 choice 7 is selected. Each time the user is prompted for a choice make sure that a number from 1 to 7 is entered, otherwise keep prompting until a valid number is entered. (Ensure that the user can change any of the choice 1 to 6 on figure 2).

- When option 3 (in the main menu shown in figure. 1) is entered, prompt the user to enter desired maximum bowling average. This is the deliveries a cricketer must bowl on average to claim a wicket. You then need to display the information of all the available Cricketers with bowlingAvg value less than the specified value. (Hint: You may use a static method called findCricketersBy, which accepts a float for a bowlingAvg and perform the needed search).
- When option 4 (in the main menu shown in figure. 1) is entered, prompt the user to enter two values, namely, strike rate and economy rate. You must display all the Cricketers who have a better strike rate and lower economy rate than the user entered values. Strike rate is the ratio of runs scored over deliveries faced, and the economy rate is the ratio of runs given over the number of overs bowled. (Hint: You may use a static method, for instance called findAllRounders, which accepts two float values, namely, strike rate and economy rate, and performs the 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 and last name as file name. The submission itself must also contain your name(s) and student ID. Use your "official" name only – no abbreviations or nick names; capitalize your "last" name. Inappropriate submissions will be heavily penalized.

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.

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 Cricketer)	**4 pts**
Default & other 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