Dept. of Computer Science & Software Eng., Concordia University
COMP 6481 --- Winter 2022

Programming and Problem Solving

Assignment 2 --- Due Wednesday, March 23, 2022

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, $n \geq 1$, and an integer m, write an algorithm as a **pseudo code** (not a program!) that would find all the flipped pairs from the given array. A flipped pair is governed by the equation b = a + m. For instance, given [1, 3, -5, 9, -2, -4, 2, 7, 4, 6], and m=3 the algorithm will return [1, 4], [3, 6], and [-5, -2]. You must only print pairs where index of a is less than that of b.

b) Find a simple solution with time complexity of $O(n^2)$

c) Can this task be performed in $O(n\log n)$ or $O(\log n)$? If yes, please provide the pseudo code for your approach. If not, explain why it is not possible.

d) Is a solution in O(n) even a possibility? If yes, please provide the pseudo code for your approach. If not, explain why it is not possible.

Question 2

Given a collection of n numbers, write an algorithm, **using pseudo code** that will output all possible combinations of r numbers that sum up to a prime number. For instance, given {1,2,3,4,5}, algorithm must output (1,2), (1,4), (2,3), (2,5), and (3,4) for $r = 2$.

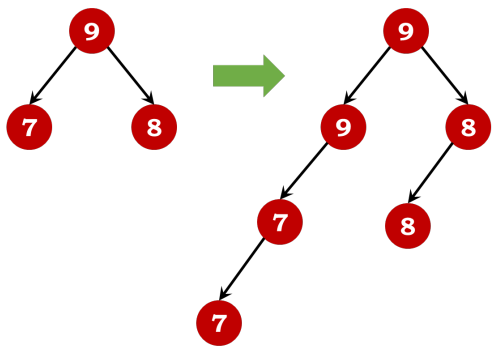
a) Write an iterative as well as a recursive solution for this problem.

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

c) What changes will you make to the algorithm to output all permutations instead. The output for the same input will now contain (1,2) as well as (2,1).

Question 3

Develop a **well-documented pseudo code** that inserts a duplicate node as the left child of the node. A sample tree transformation is provided below.



Is it guaranteed to have the resulting tree as a balanced tree (BT) if the input was a BT?

Part II

Purpose: The purpose of this assignment is to allow you practice Exception Handling, and File I/O, as well as other previous object-oriented concepts.

You will have to design a **Sales** class with the below mentioned attributes: a *country* (String type), an *item_type* (String type), an *order_priority* (Char type), an *order_date* (Date type), an *order_ID* (long type), a *ship_date* (Date type), a *units_sold* (Int type), a *unit_price* (Float type), a *unit_cost* (float type), a *revenue* (Double type), a *total_cost* (Double type), and a *total_profit* (Double type). Date fields will be in dd/mm/yyyy format.

For this assignment, you are required to:

1. Write the implementation of the Sales class according to above given specification.
2. Write implementation of an exception class called DuplicateRecordException, which extends the Exception class. Should a duplicate record be detected at any point, an object from this class will be thrown.
3. Write the implementation of a public class, called SalesDatabase, which will utilize the Sales class and the files provided, as explained below. The class will have a static array of Sales, called salesArr[], and few methods. Beside the main() method, this class must have the methods mentioned further down.

Firstly, you will have to list the directory and file structure at a given path and then you will have to process all the files present at that path. A detailed task description is given below:

1. List all the directories, subdirectories and files present at a path. You should assume that there is a folder named Data which will be available in the same directory as your .java file and will contain all the files that you must process. You should open a file named "log.txt" and write the detailed file structure present in the Data folder in it. You must take the recursive approach to list all files and directories present in the Data folder. For example, a sample log of directory system is shown in figure 1. You must use exception handing while using the File class.

File Edit Format View Help

```
directory:c:/EclipseWorkspace/Assignment2/Data/1
     file:c:/directory:C:/EclipseWorkspace/Assignment2/Data/1/Tom.txt
directory:c:/EclipseWorkspace/Assignment2/Data/2
     file:c:/directory:C:/EclipseWorkspace/Assignment2/Data/2/Sam.txt
directory:c:/EclipseWorkspace/Assignment2/Data/3
     file:c:/directory:C:/EclipseWorkspace/Assignment2/Data/3/Harry.txt
```

Figure 1. A sample directory structure and corresponding log file

2. The program should provide three options to users. (1) List files, (2) Process files, and (3) Exit. First option will create the "log.txt" file and second option will open "log.txt" and all the files logged in will be processed as described below. If a file named "log.txt" is already present, you must overwrite its contents. The third option will exit the program. You MUST however, close all opened files before exiting.
3. Write 2 custom exception classes InvalidFileException and EmptyFolderException. The first one responsible to make sure the log file is valid and the file paths in it are valid. The second one will be responsible for checking for empty directories. They must have constructors to allow a default error message "Error: Input file named XXX cannot be found. They should also allow for a custom message to be set as error message. All the exceptions should be logged and the files which are valid must be processed.

4. Processing phase of this application will iterate through the log file, if it exists, in the following manner:
   a. Go through each subdirectory inside folder named "Data" and process text files within each of the subdirectories.
   b. You must identify the duplicate records and notify the user of these records. Moreover, you must not put duplicate records in the output database. A sample output file is provided along with this assignment.
   c. SalesDatabase will have the below mentioned methods to facilitate processing of data in an effective fashion.
      - A method called addRecords(), which accepts one parameter: a Sales object; and adds it to the salesArr[] object.
      - A method called displayFileContents(), which accepts an input file stream name, then displays contents of this file to the standard output. This method however must use the BufferedReader class to read the file.
      - A method called binarySaleSearch(), which accepts one parameter: an *order_ID* that is used to search a record using the binary search approach. The method must also keep track and display how many iterations it needed to perform the search.
      - A method called sequentialSaleSearch(), which takes 1 parameter: an *order_ID* that is used to search a record using the sequential search approach. The method must also keep track and display how many iterations it needed to perform the search.
      - You can add any other methods to that class if you wish to. You may make use of built-in functionalities to sort the array for binary and sequential search.
5. Finally, here are some general information:
   a. It may assist you greatly if you take advantage of static variables/attributes and static attributes throughout the assignment; in fact, it is not necessary to utilize other aspects such as Inheritance, Polymorphism, etc.
   b. You must use recursion for creating file structure log.
   c. You should minimize opening and closing the files as much as possible; a better mark will be given for that.
   d. Do not use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!
   e. Again, your program must work for any directory structure if the root directory is present in the same directory as your .java file and is named "Data". The files provided with this assignment are only one possible version and must not be considered as the general case when writing your code.

SUBMISSION INSTRUCTIONS

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/nick names; capitalize the usual "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 (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!

Total	**10 pts**
Documentations	**1 pt**
JavaDoc documentations	1 pt
Tasks	**9 pts**
Task#1	2 pts
Task#2	1.5 pts
Task#3	1.5 pts
Task#4	4 pts