Unity University Adama Campus
Department of Computer Science
Data Structures and Algorithms (CoSc3061)

Project

Instruction: Read and choose one of the following specification and write a C++ program which implements the given. The tasks that your program performs should be divided in different functions according to their similarity. Furthermore, your program should be readable and well commented. Make a group 5 students and all the members should work and present the project together.

Type

Submission: Submit and present the source code one week before final exam, before January 15, 2024

1. ***Book List***

   Assume that a Library needs automated system that manages the purchase of books. Every book has *ISBN*, *Author*, *Title*, *Publisher*, *Publishing Date*, and *Number of pages*, and *Total Quantity*. The routine operations commonly done by the librarians on the books are searching a book using different parameters (such as, ISBN, title, author), sorting books by one of the two parameters (ISBN or Title), updating book information (such as total quantity of a given book whether by adding or decreasing), adding new book in the library system, removing a book completely from library system. Based on the above description, do the following:

   a. Write a formal specification (both attributes and operations) of an ADT called "BookList".
   b. Write a C++ program which implements ADT "BookList" specified in question 1(a) above using a doubly-linked list. At least your program should implement the following operations.
      - Add book at:
        - The beginning of the list
        - The end of the list
        - Any place assuming the list is sorted based on Title
        - Any place assuming the list is sorted based on ISBN
      - Traverse and display all the books in "BookList" from first to the last.
      - Search a given book using "Title", "ISBN" or "Author"
      - Update a given book's detail
      - Delete a specific book from "BookList"

   NB: Use the following as guidelines or hints while implementing the "BookList":

   - ✓ Create an appropriate self-referential structure by including all the attribute
   - ✓ Make sure the BookList is initially empty (i.e. head must be NULL)
   - ✓ The only external pointer is "head" pointer (do not add "tail" pointer)
   - ✓ Your program should be menu driven

2. *Guess Number Game*

Let us consider a typical game played by kids. You are asked to guess the number thought of by your friend in the range of 1 to 100. You are to guess by asking a minimum number of questions. Of course, you are not allowed to ask the number itself. The easiest approach is to start asking, 'Is it 1?' In case the answer is 'No', then ask, 'Is it 2?' Continue this process in the ascending order of integers till you get the answer as 'Yes'. What if the number your friend has in mind is 99? Obviously, this approach is not an efficient one and also it is boring to ask too many questions. The solution to this problem is to ask a question, 'Is it 50?' If no, another question to be asked is, 'is it greater than 50?' If the answer is 'Yes', then the range to be searched is 51 to 100, which is half of the previous range. If the answer is 'No', the range is 1 to 49, which is again half of the original. You may continue doing so till you guess the number. Surely, the second approach reduces the total number of questions asked on an average. Modify the above game to be played by more than one kid at a time, the program should keep track of each player and finally it will congratulates the player which has guessed the number with a minimum number of trial (question) and minimum total difference at each guess.

E.g., if 84 is the number thought by the program, then the two players (player A and B) may respond as following:

Previous turn: NO
Question of player A: >=50?
Answer of the computer: Yes
Guess of player A: 80
Wrong!
(The program should clear the output screen after the current player finishes his turn)
Previous turn: NO
Question of player B: >=10?
Answer of the computer: Yes
Guess of player A: 20
Wrong!
(The program should clear the output screen after the current player finishes his turn)
Previous turn: Question of player A: >=50?
Answer of the computer: Yes
Guess of player A: 80
Question of player A: >=70?
Answer of the computer: Yes
Guess of player A: 90
Wrong!
(The program should clear the output screen after the current player finishes his turn)
Previous turn: Question of player B: >=10?
Answer of the computer: Yes
Guess of player A: 20
Question of player A: >=80?
Answer of the computer: Yes
Guess of player A: 84
Right! Congratulation your guessing performance is 98.92%

(The program should clear the output screen after the current player finishes his turn)

Previous turn: Question of player A: >=70?
Answer of the computer: Yes
Guess of player A: 90

Question of player A: >=85?
Answer of the computer: No
Guess of player A: 84
Right! Congratulation your guessing performance is 98.92%
GAME ENDS
Player A won with guessing performance of **98.37%**
Player B losses with guessing performance of **98.16%**

TD (total differences) = $\sum$ | Thought Number- Guessed Number at each turn |
PT (performance in trial in percent) =
100 - ((Total number of trials/maximum number of trials) X 100)
PA (performance in accuracy in percent) =

$$100 - \left(\left(\frac{TD}{\sum_{i=1}^{n} i - ((ThoughtNumber - MaximumNumber) * ThoughtNumber + ThoughtNumber)} \right) * 100 \right)$$

Where, n is the maximum (largest) number

Guessing Performance = (PT + PA)/2

Therefore, write a C++ program which simulates the second approach. Your source code should be readable and modular.

NB: You should use at least two data structures in your program

3. Phone Management Systems using doubly linked list

Phone management systems allows users to create contacts list in their phone directory. Once they create a list of contacts, user can perform the following operations on it.

1. Add new contact to the list
2. Display contact list/ phone book
3. Update details on existing contact (name, phone number or email)
4. Delete contact
5. Delete same name in phonebook
6. Delete same numbers in phonebook
7. Search for a contact in phone directory. The search query on a string 'bet' displays all the contacts which prefixes as 'bet'. One special property of the search function is that when a user searches for a contact from the contact list then suggestions are shown after user enters each character.

Remarks: use lowercase alphabets while create a contacts list.

When you start running your program, it starts as follows:

What is your name? Hiwot

Welcome Hiwot##########

Please enter the following info to create your phonebook:

Enter your contact name: Bethelehem
Enter phone number: 251912….
Enter your contact e-mail: beth@gmail.com

Do you want to continue? Y/y for(yes)
It will keep allowing you create a contact for other person.
Enter your name: Kalkidan

Enter phone number: 251911….

Enter your e-mail: kal@gmail.com

Do you want to continue? n/N(no)

1. Display your phone book
2. Insert new contact
3. Update details on existing contact
4. Delete contact
5. Delete same name in phonebook
6. Delete same numbers in phonebook
7. Search

If you select option 1, it will display each contact in the contact list one by one on newline as follows:

Name	Phone Number	E-mail
Bethelehem	251912….	beth@gmail.com
Kalkidan	251911….	kal@gmail.com

Do you want to continue operations?Y/y for (yes)

1. Display your phone book
2. Insert new contact
3. Update details on existing contact
4. Delete contact
5. Delete same name in phonebook
6. Delete same numbers in phonebook
7. Search

If you select option 2, it will allow you to add a new contact as follows:

Enter your contact name: Kalkidan

Enter phone number: 251911….

Enter your contact e-mail: kal@gmail.com

Do you want to continue operations? Y/y for (yes)

1. Display your phone book
2. Insert new contact
3. Update details on existing contact
4. Delete contact
5. Delete same name in phonebook
6. Delete same numbers in phonebook
7. Search

4. Cinema ticket booking system

This system provides convenient way for a customer to buy cinema tickets. It should include the following functionalities for the user

1. viewing the timings of different movies (take any cinema system you know,i.e. Alem cinema)

   For example:

   Movies:
   1. Yewondwoch guday
   2. The crown
   3. Semayawi Feres

   Enter your movies of choice:
   Let say you take choice 1(Yewondowch guday) then it will show the user all the possible times , he can watch.

   Enter time to watch: 1…

   Then it will ask seat types (VIP, normal):

   Once the user decided the seat types, it will show him all possible free seats and a user can pick one. If all the seats are occupied, it shall the user to change his watch time.

   Operations:
   1. Create cinema

2. Add movies
3. Book ticket
4. Change schedule
5. Display ticket information

To implement this project you need to use a queue (priority queue) and implement it using linked list