**National University of Computer & Emerging Sciences**
**Peshawar Campus**

**Student Name:** _____ **Roll No:**_____

**Program:** CS+SE+AI **Examination:** Final

**Semester:** Fall 2023 **Total Marks/Weightage:** 55

**Time Allowed:** 02 hours 30 minutes **Instructor:** Fariba Laiq

**Course:** Data Structures **Date:** 12th Dec 2023

**NOTE:**

- Attempt all questions.
- Don't rush, carefully understand the problem and then go for solution.
- Understanding the question is the part of Exam.
- An individual may be assigned a straight-forward 0 if the submitted assessed task is copied/cheated from another individual/internet/found using internet or chatGPT.
- Create a folder named as Roll number, name and section, copy all the files (Q1.cpp, Q2.cpp, ….), in that folder and submit the folder.

---

## Note:

You will only be given marks for filling the unimplemented part. As the rest of the code is already provided to you.

You are given the template codes that contains the partially implemented structure of the program. So, you are not required to write the code from scratch. Simply implement the unimplemented parts in the template provided in the same folder.

# Haunted Mansion Exploration

### Implement Queue using array and Stack using linked list

Weightage [20]

In the heart of a mysterious forest, hidden away from the city, there exists a Haunted Mansion, having restless spirits and lost treasures. Brave paranormal investigators choose to explore this mansion, guided by the glow of their lanterns. The exploration of this

mansion is fraught with supernatural challenges, and both queues and stacks are essential for their survival.

The queue contains the order of the places to explore in the mansion, that leads from the entrance till the exit. That will be already stored in the queue when program starts.

The stack will store the records of the most recent ghost encounters.

Otherwise, the explorer might get trapped in the dead end of the mansion.

The queue contains the order in the following way.

- The Gateway to Darkness (the entrance)
- The Creeping Corridor
- The abandoned lounge
- The haunted gallery
- The shadowed library
- The broken window (the exit)

Now while exploring any place, there is a chance that you will encounter a ghost, or an evil spirit if so, keep the record of that encounter in the stack, so that when you exit from the mansion, you can tell your head about the most recent ghost encounters, along with their place.

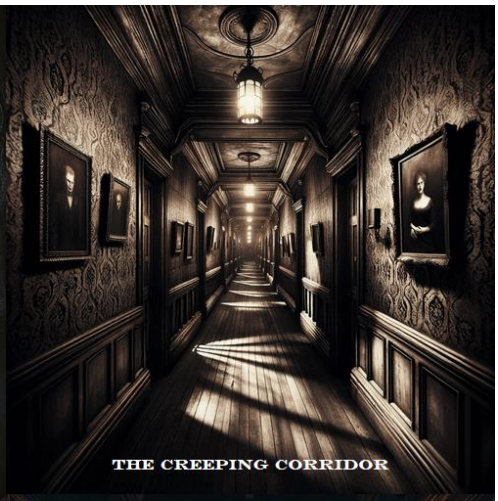
The program should be menu driven, when the program starts, the user can start exploring the mansion places one by one, after that he will explore the places in order according to the queue (note that the queue is already filled with the names of the places when program starts, user will not be going to enter the places names, the program will dequeue and display it on the screen). When he explores all the places of the mansion and left from it, simply print a message “You left safe and sound!” and display all the places where he encounters an evil spirit.

Implement the un-implemented methods and the main function. Make sure to write all the checks with each data structure.

Use your mind when to call which method of each data structure.

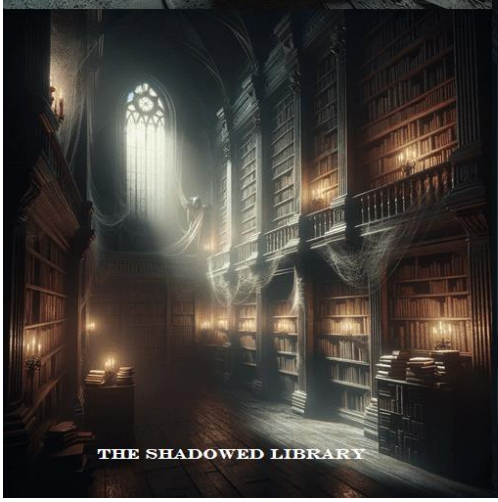
Below is a depiction of the places of the mansion for a better imagination (generated by GPT-4).


THE GATEWAY TO DARKNESS


THE CREEPING CORRIDOR


THE ABANDONED LOUNGE


THE HAUNTED GALLERY


THE SHADOWED LIBRARY


THE BROKEN WINDOW (THE EXIT)

**Sample Output:**

```
Start exploring the mansion?
1. Yes
2. No
1
Currently at: The Gateway to Darkness
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
2
Currently at: The Creeping Corridor
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
1
What did you see?
A moving shadow
Currently at: The Abandoned Lounge
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
1
What did you see?
A moving chair
Currently at: The Haunted Gallery
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
2
Currently at: The Shadowed Library
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
1
What did you see?
A falling book
Currently at: The Broken Window
Did you encountered a ghost or an evil spirit?
1. Yes
2. No
2
Yayyy, you escaped safe and sound!
Most recent ghosts/evil spirits encounters:
A falling book at: The Shadowed Library
A moving chair at: The Abandoned Lounge
A moving shadow at: The Creeping Corridor
```

**Employee Performance Evaluation Program**

Weightage [20]

Develop a C++ program that uses a max heap data structure to rank employees based on their performance scores. The program should input employee data, calculate their scores based on predefined criteria, and use a max heap to store them in descending order of performance score.

Implement a structure or class Employee with relevant attributes such as employeeID, name, and performance metrics (e.g., percentage score in sales, customerSatisfaction, projectCompletion), and the overall performance score. All of these score represents the values in percentage.

Define a scoring function within the class to calculate the performance score of each employee based on these metrics.

Performance Score = sales+customer satisfaction+project completion

The program will be menu driven where the user will have the choice to:

1. Enter a new employee details
2. Search an employee against a specific performance score.
3. Get the top-ranking employee
4. Update an employee score
5. Exit

When the user selects the get top ranking employee, the program should display the top-ranking employee among the current employees and after that remove it from the tree. Similar action will be performed when the user wants to get the top-ranking employee again. The program should display the top-ranking employee among the employees that are present in the tree currently.

When the user wants to update an employee score, the program should ask the new values for all of three scores, and update the info and also correctly places the node in the tree.

**Sample Output:**

```
Menu:
1. Enter a new employee details
2. Get the top-ranking employee
3. Update an employee's scores
4. Display employee rankings
5. Exit
1
Enter Employee ID: 100
Enter Employee Name: Fariba
Enter Sales (in percentage): 90
Enter Customer Satisfaction (in percentage): 80
Enter Project Completion Efficiency (in percentage): 70
Employee added successfully!

Menu:
1. Enter a new employee details
2. Get the top-ranking employee
3. Update an employee's scores
4. Display employee rankings
5. Exit

1
Enter Employee ID: 200
Enter Employee Name: Ali
Enter Sales (in percentage): 60
Enter Customer Satisfaction (in percentage): 70
Enter Project Completion Efficiency (in percentage): 50
Employee added successfully!

Menu:
1. Enter a new employee details
2. Get the top-ranking employee
3. Update an employee's scores
4. Display employee rankings
5. Exit
1
Enter Employee ID: 300
Enter Employee Name: Hina
Enter Sales (in percentage): 40
Enter Customer Satisfaction (in percentage): 60
Enter Project Completion Efficiency (in percentage): 70
Employee added successfully!
```

```
Menu:
1. Enter a new employee details
2. Get the top-ranking employee
3. Update an employee's scores
4. Display employee rankings
5. Exit
4
Employee Rankings:
Employee ID: 100, Name: Fariba, Performance Score: 240
Employee ID: 200, Name: Ali, Performance Score: 180
Employee ID: 300, Name: Hina, Performance Score: 170

Menu:
1. Enter a new employee details
2. Get the top-ranking employee
3. Update an employee's scores
4. Display employee rankings
5. Exit
2
Top-ranking employee: Fariba with performance score: 240
```

**Movie Recommendation System**

Weightage [15]

Develop a movie recommendation system that combines binary search trees and an array with any sorting algorithm of your choice, to provide personalized movie recommendations to users.

Create a menu-driven program that has the following choices:

1. Add a movie
2. Delete a movie
3. Recommend movies
4. Display Movies

When the user wants to add a movie, the program should ask the following info:

- Movie ID
- Title
- Genre
- Ratings

The movies will be stored in the BST on the basis of the movie ID.

The movies will also get stored in an array (make a fixed size array of 5 elements). When the user selected display movies, the program actually displays the sorted array and show movie names along with their ratings in descending order.

When the user selects recommend movies, the program should ask the user, about his favorite genre, the program will display all the movies of that genre.

When a user selects delete a movie, the program should delete the movie by ID or name from the tree.

**Note:** In this task, you will be given a templated that has partially implemented code, but has some errors too, you are supposed to fix the errors and adjust to code to the requirements, and complete the template by implementing the unimplemented methods.

**Sample Output:**

Imagine it yourself this time! 😊

**BEST OF LUCK!**