Assignment 1 CSCI112 Fall 2023 10 points

Due on: **Sunday September 10th -1100PM**

Objective:

The objective of this task is to create and implement a class for controlling an automated elevator based on a pre-defined design. The simulation of real-world activities, such as controlling traffic lights or supermarket checkouts, is a common application of computers. This assignment serves as an introduction to simulating and controlling an automatic elevator using random numbers to emulate natural occurrences.

Procedure:

To accomplish this task, you are required to create the Elevator class, which represents the physical entity of the elevator. The elevator operates in a building with floors numbered from 0 to maxfloor. It remains stationary without any requests. There are two sets of buttons that the elevator responds to: floor buttons and internal buttons. When a request is made from a floor button, the elevator is activated and moves towards that floor. Upon reaching the floor, the doors open for riders to enter and select the desired floor on the internal button pad. If no further requests are made, the elevator proceeds to the requested floor. The elevator continues to travel in the same direction if there are requests for that direction. Once it reaches the last requested floor, it attempts to satisfy requests in the opposite direction.

The Elevator class, described partially below, includes the necessary member functions and variables:

```
//
//  Elevator.h
//  ElevatorSim
```

```
//
//  Created by Sherine Antoun on 4/21/23.
//  Copyright © 2023 Sherine Antoun. All rights reserved.
//  You may not alter this header file in any way.
/*
#ifndef Elevator_h
#define Elevator_h
#include <stdio.h>
const int UP = 1, DOWN = -1, STOPPED = 0;

class Elevator
{
public:
   // Constructor for the number of floors - no default constructor
   Elevator(int);
   ~Elevator(); // Destructor
   void Move(); // Move one floor in the current direction
   // Report current status: floor, direction, door status
   // handle rider request outside elevator
      void DirectionSelect(int, int);
   // handle rider request inside elevator
      void ChooseFloor(int);

   private:
      int maxfloor;    // highest numbered floor in the building
      int currentfloor;
      int currentdirection;
      int dooropen;
```

```
        int* floorup;    // array for up buttons on floor
        int* floordown;    // array for down buttons
        int* button;    // array of internal buttons
};
*/
```

To facilitate communication between the elevator and the outside world, public member functions are added to access the buttons. The `DirectionSelect(int, int)` function allows a rider at a specific floor to select an up or down request, while the `ChooseFloor(int)` function allows a rider inside the elevator to select the desired floor. The elevator operates even if a rider does not enter, ensuring that the buttons are turned off when the elevator reaches a floor where the doors open.

The assignment includes the header file `Elevator.h` and a driver program `Classprac.cpp` that provides a simulation of requests to the elevator, enabling you to test your class implementation.

Task:
Your assignment entails supplying and coding within the `Elevator.cpp` file. This file should encompass the complete implementation of all member functions and variables essential for the elevator's functionality. It's crucial to understand that the design of the `Elevator.h` file must remain unaltered; therefore, no modifications to it are permitted.

Additionally, you are expected to furnish a makefile capable of building the project using the clang++ compiler exclusively. Please note that for grading purposes, only clang will be considered; no other compilers will be entertained.

The grading process will adhere to the criteria outlined in the rubric accessible in the miscellaneous documents folder on D2L.

Kindly be aware that submissions submitted after the deadline will not be evaluated.

Moreover, any submissions incorporating STL constructs or functions will not be accepted.

Submissions that fail to compile using the makefile you provide will be ineligible for grading. However, you will receive constructive feedback.

Please be advised that I won't be able to respond to email inquiries related to this task within the 48-hour period preceding the submission deadline.