

**Zewail City of Science and Technology**

**C++ Programming Lab**

**(NANENG 312)**

**Spring 2021**

**C++ Course Project**

### Introduction

You are to design the software for an elevator embedded system. Your code should support multiple elevators per floor. Each of them would respond to elevator requests according to a priority system.

You should work in pairs. Plan on paper so you would know what instructions to code. Divide your code into functions for you to implement and distribute the workload evenly. Do not start coding a function until you are 100% sure what it should do and how it interacts with the rest of the code in terms of inputs, implementation and outputs.

### System description

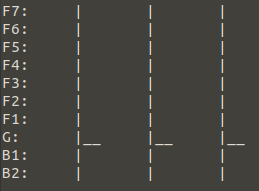
There are 7 floors above the ground, each labeled “F1”, “F2”.. “F7”; a ground floor, labeled “G”; and two basement levels, labeled “B1” and “B2” with “B2” being the lowest level.

Each elevator can stay stationary, close the door, go up a floor or go down a floor in each time step. An elevator should have its doors closed at all times except when it reaches a destination floor or when allowing a requester to enter. In that case, it instantly opens the door for a time step, closes the door for a time step, then proceeds to its next floor (or stays where it is if there are no elevator requests it needs to handle.

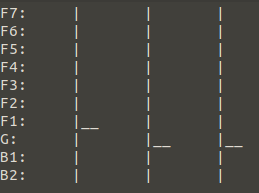
### Elevator visuals

Consider the following elevator system:

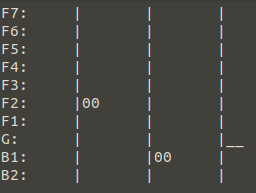
at time step 0, all elevators are at ground level. A request-to-go-up is made at F3.



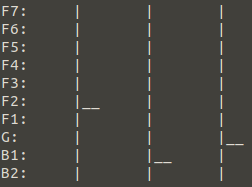
At time step 1, a request-to-go-up is made at each of F2 and B1. The leftmost elevator continues to move upwards and the middle one starts to move downwards.



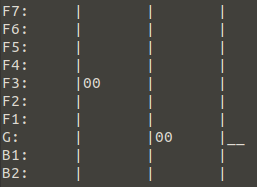
At time step 2, no requests are made. The leftmost elevator stops and instantly opens the door for the requester in F2. She chooses F7. The middle elevator reaches B1 and instantly opens the door. Its requester chooses G.



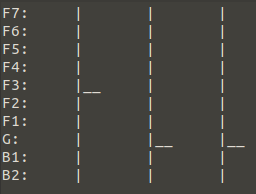
At time step 3, both elevators stay stationary as they close the door.



At time step 4, the leftmost elevator goes up and instantly opens the door for the requester from F3. He chooses F6. The middle elevator reaches its destination and instantly opens the door for the client to exit.



At time step 5, both elevators close the door. The leftmost one will move to its destination in the following time steps. A request-to-go-down is made on F5 (the client will press on G when she enters the elevator in the future). The middle elevator should go to it in the following time steps.



If an elevator is open at a floor for a client to get out and another client wishes to enter during the same time step, the elevator does not wait. Each client can enter or exit instantly but the elevator has to spend the following time step stationary with its door closed.

If the door is closed during a time step and a client requests an elevator at the same time instant, the door opens in the following time step and spends the next time step closed and stationary.

### Requests

Requests are to be made through the console (using cin and cout), and console input should be prompted once at every time step.

The console input for the request at time step 0 in the example above would be:

F3-U-F6.

The requests at time step 1 would be:

F2-U-F7 B1-U-G.

The requests at time step 5 would be:

F5-D-G.

### Priority Rules

1. If a client sends a request to go up, and an elevator is already going up and will pass by (or stop at) the requester’s floor, no other elevator is called. The ascending elevator should take the request and stop at the requester’s floor. This holds true for descending elevators and clients who wish to go down.
2. If a client sends a request to go down and an ascending elevator shall pass by his floor, the elevator should not take the request. Another elevator should be called instead. This holds true for clients going up and descending elevators.
3. If a request is made and no moving elevators are allowed to take it due to rule 2, the nearest elevator should take the request.
4. If a request is made and a moving elevator shall pass by the requester’s floor on its way, and the requesting client is moving in the direction of the elevator, your code should check if another elevator is less than 3 floors away. If a nearby elevator is closer than the moving elevator, the nearby elevator takes the request. Otherwise, the nearby elevator is not called and the moving elevator should make a stop at the client’s floor.

### Phases

Your work will be delivered in two phases. In phase 1, you should have implemented the graphical illustration for the elevator system. It should be similar to the one in the screenshots above. You should also deliver a class file for the elevator class and another for the client class. You should also prepare function implementations in a header file which order elevators to move up and down depending on clients’ requests.

In phase 2, you should deliver your work after implementing priority rules. We will probably make you read the client requests from a file instead of user input to make it easier. But until we tell you to do so, you only have to submit priority rules for phase 2.

## Note

Plagiarism will not be tolerated. Your code has to be 100% original. Be prepared to modify your code if asked in the project discussion. Each team member should have full knowledge of the code. We highly recommend that you use comments wherever possible.