

Program in C++ using Threads : Simulation of Elevator Program with 9 floors

**in
Parallel and Distributed Computing (CS0051)**

**Instructor:
Doc. Hadji Tejuco**

**Members:
Gian Ijy Clemente
Jabenn P. Tinio**

27/01/2025

TABLE OF CONTENTS

III . Introduction

IV . Project Overview

V . Requirements Analysis

VI . System Design

VII . Implementation

VIII . Testing

IX . User Manual

X . Challenges and Solutions

XI . Future Enhancements

XII .Conclusion

XIII .References

XIV. Appendices

III.Introduction

A. Purpose

The purpose of the program aims to enhance our comprehension of multithreading ideas in C++ through the simulation of a real-world application: an elevator system including nine stories. This simulation aims to demonstrate how threads can be effectively used to manage concurrent tasks, such as handling multiple elevator requests, managing floor-level operations, and optimizing responsiveness.

This program aims to:

1. Explore the practical use of C++ threads and synchronization mechanisms
2. Enhance our problem-solving skills in designing systems that require parallel and distributed computing principles.
3. Build a foundational understanding of thread-safe programming to prepare for more complex applications in CS0051: Parallel and Distributed Computing.

This activity functions as both a practical exercise and a framework for implementing theoretical principles in a real-world context, along with the course objectives of promoting hands-on experience with parallelism and concurrency.

B. Objectives

We are asked to provide a simulation of an elevator program based on our own specifications. The program has several objectives, which can be pointed out as follows:

1. To create a simulation of real-world elevator functionality. To the best of the developers' skills, they replicate a near-real-world elevator function that can ascend or descend floors of a nine-story building.
2. To demonstrate multithreading in C++ by a simulation of an elevator program. Handling user requests and controlling elevator movements.

C. Scope

The scope of the program is a single user simulation of a real-world elevator.

IV. Project Overview

A. Problem Statement

To simulate an Elevator Program, with 9 floors.

B. Key Features

1. **Visualization** : Shows the user a simple text visualisation of the floor where the simulated elevator is, when it is moving, and whether it has arrived at its destination.
2. **Multiple Request Handling** : Can handle multiple floor requests from the user.

V . Requirements Analysis

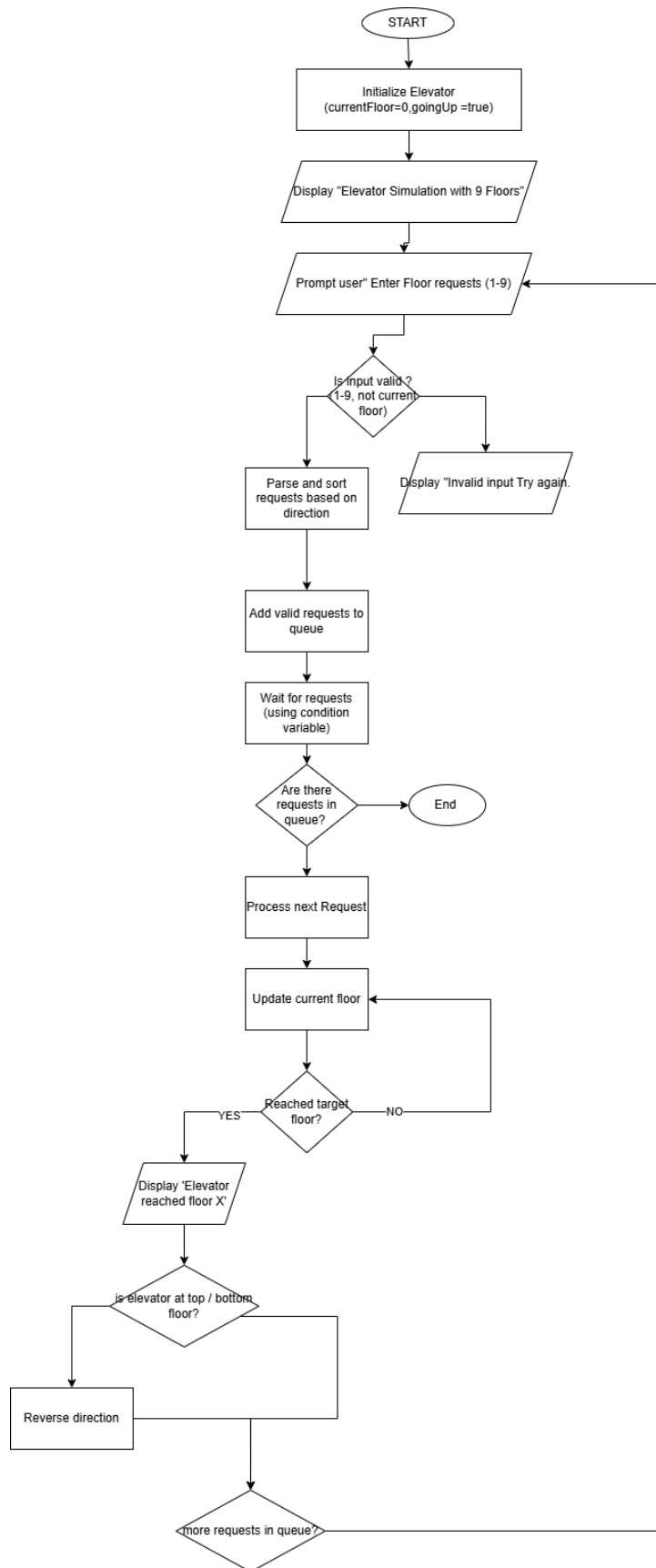
A. Functional Requirements

1. **Elevator Simulation**: Enable simulation for moving between floors (1-9).
2. **Request Processing**: Accept and process user-inputted floor requests.
3. **Request Sorting**: Sort floor requests in ascending or descending order based on the elevator's direction.
4. **Direction Control**: Reverse direction at top (9th floor) or bottom (ground floor).
5. **Real-time Status**: Display current floor during movement.
6. **Input Validation**: Validate user input for floor requests

B. Non-Functional Requirements

1. **Performance**: Minimal response time for processing and moving between floors.
2. **Concurrency**: Efficient handling of requests using multithreading.
3. **Security**: Thread-safe access to shared resources
4. **Usability**: Clear prompts and instructions for user input, handling invalid inputs gracefully.
5. **Scalability**: Extendable to support more floors or multiple elevators.

VI . System Design



VII . Implementation

A. Technologies Used

1. Programming Languages :
 - a. C++ 17
2. Frameworks and Libraries:
 - a. C++ Standard Library (STL)
 - b. <thread> -
 - c. <mutex>
 - d. <condition_variable>
 - e. <queue>
 - f. <vector>
 - g. <algorithm>
 - h. <string>
 - i. <chrono>
 - j. <iostream>
3. Tools:
 - a. OnlineGDB

B. Screenshot:



The screenshot displays the OnlineGDB C++ IDE interface. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, Beautify, and a user profile icon. The language is set to C++ 17. The editor shows the following code in `main.cpp`:

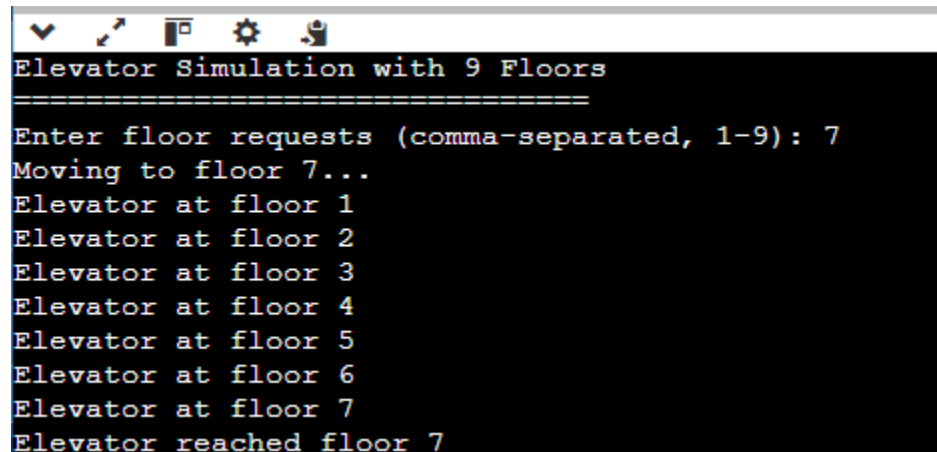
```
118 };
119
120 int main() {
121     cout << "Elevator Simulation with 9 Floors\n";
122     cout << "=====\n";
123     Elevator elevator;
124     elevator.start();
125     return 0;
126 }
```

The output window shows the execution results:

```
Elevator Simulation with 9 Floors
=====
Enter floor requests (comma-separated, 1-9): 9
Moving to floor 9...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9):
```

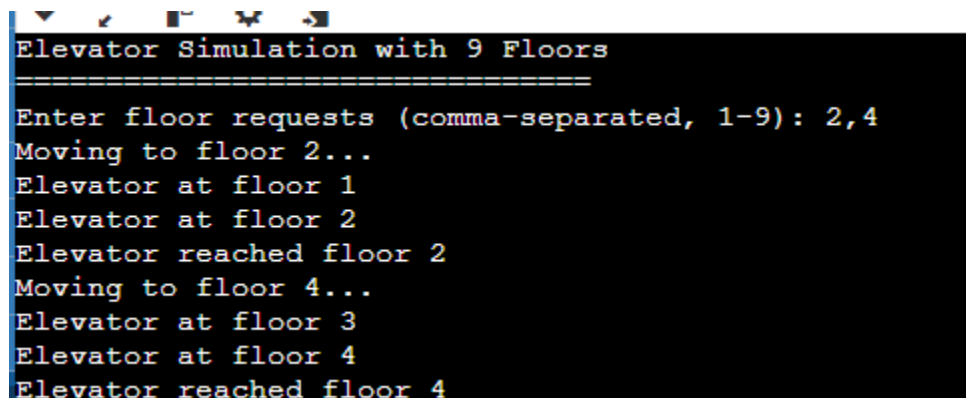
VIII . Testing

a. Single Input - choosing 1 floor



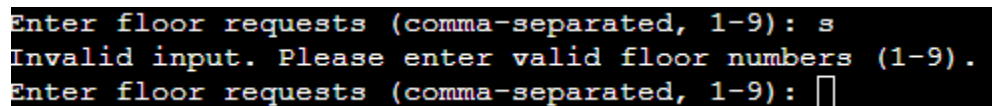
```
Elevator Simulation with 9 Floors
=====
Enter floor requests (comma-separated, 1-9): 7
Moving to floor 7...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator reached floor 7
```

b. Multiple inputs - choosing multiple floors



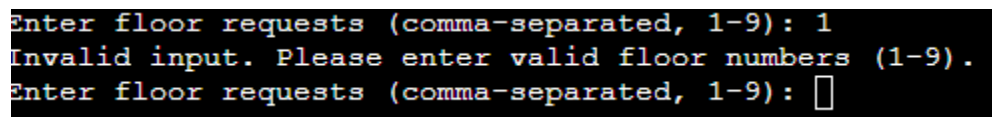
```
Elevator Simulation with 9 Floors
=====
Enter floor requests (comma-separated, 1-9): 2,4
Moving to floor 2...
Elevator at floor 1
Elevator at floor 2
Elevator reached floor 2
Moving to floor 4...
Elevator at floor 3
Elevator at floor 4
Elevator reached floor 4
```

c. Inputting a non-numerical value



```
Enter floor requests (comma-separated, 1-9): s
Invalid input. Please enter valid floor numbers (1-9).
Enter floor requests (comma-separated, 1-9): 
```

d. Inputting the same floor value



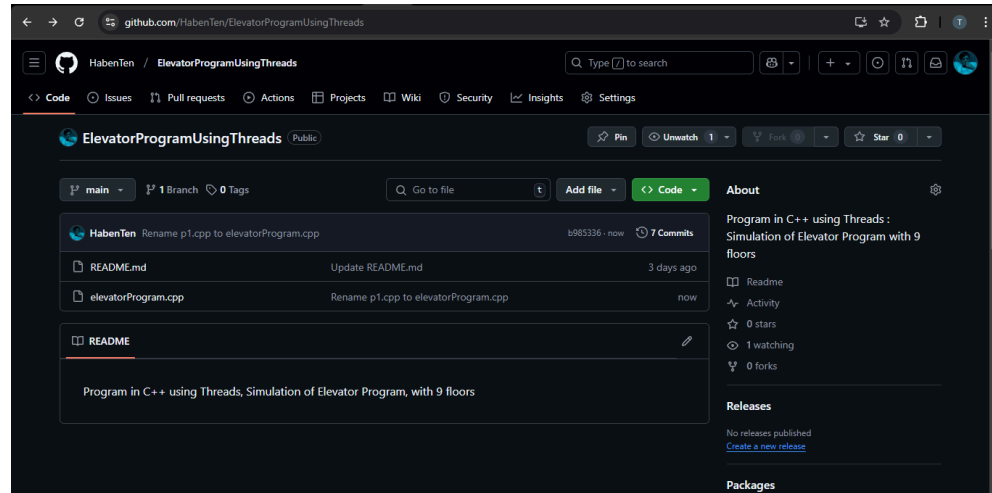
```
Enter floor requests (comma-separated, 1-9): 1
Invalid input. Please enter valid floor numbers (1-9).
Enter floor requests (comma-separated, 1-9): 
```

IX . User Manual

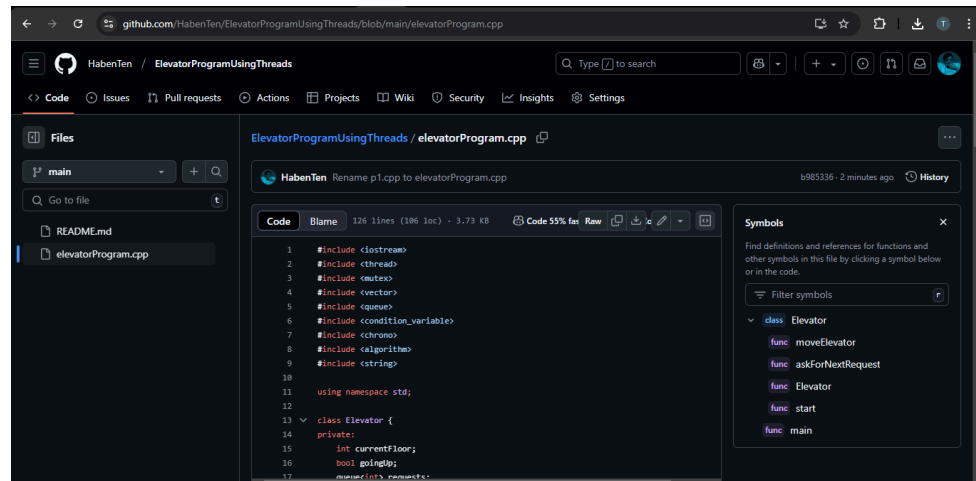
A. Installation Guide

- a. Visit our git repository at

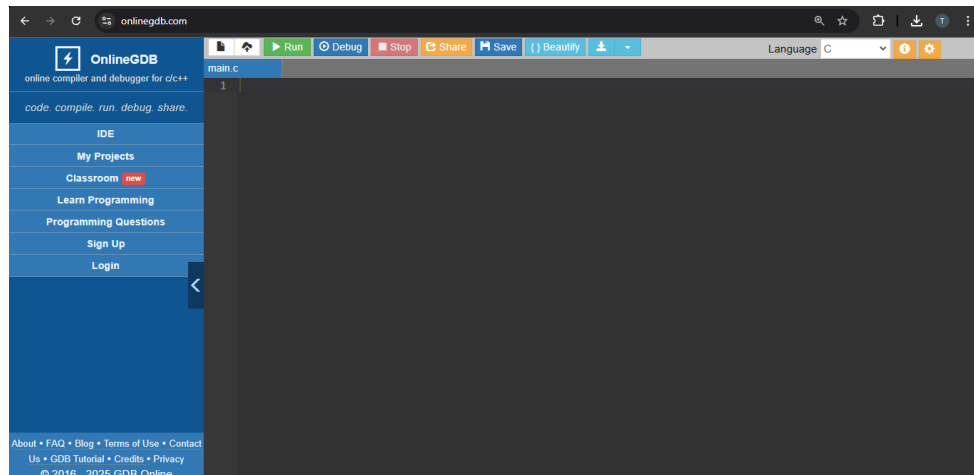
<https://github.com/HabenTen/ElevatorProgramUsingThreads.git>



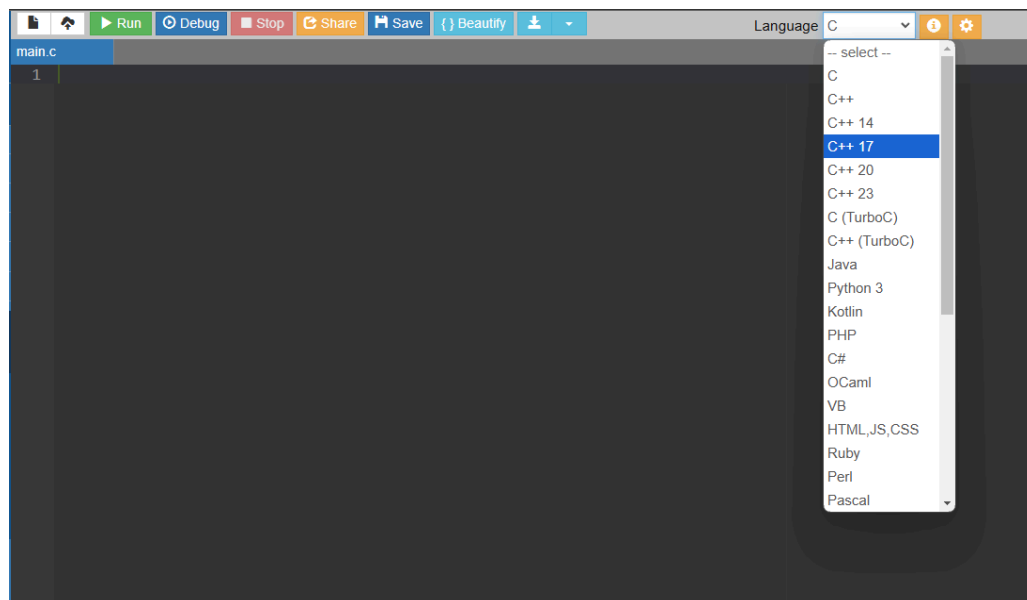
- b. Click elevatorProgram.cpp, either download it or copy the code. However for this tutorial we will be using OnlineGDB for ease of use, so we recommend to copy the code.



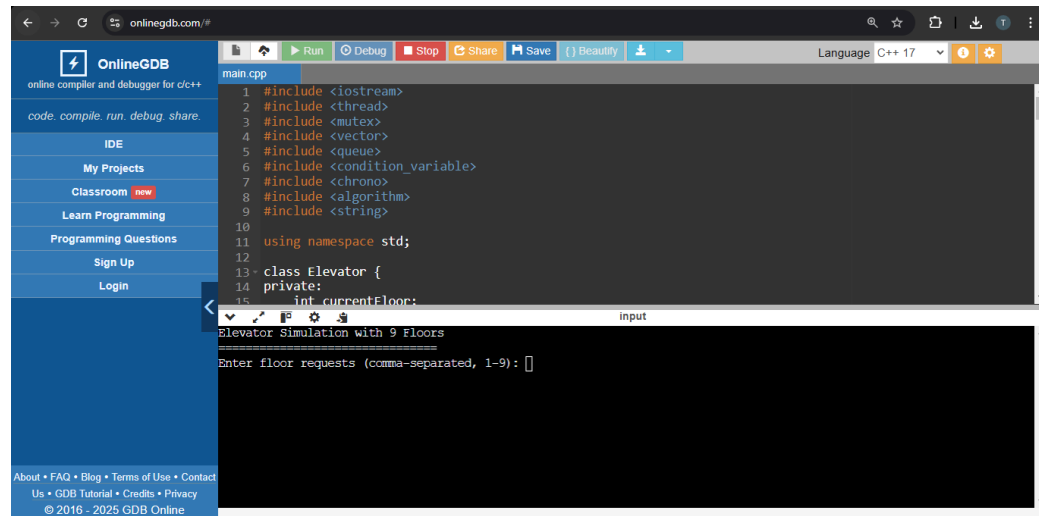
- c. Once you copied the code go to [onlinegdb.com](https://www.onlinegdb.com)



- d. On the top left you will see Language with a drop down selection box, click on it and then choose C++ 17.



- e. Simply paste the code on the working environment then run.



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code.compile.run.debug.share, IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. At the bottom of the sidebar are links for About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, and Privacy, along with the copyright notice © 2016 - 2025 GDB Online. The main editor area displays a C++ file named 'main.cpp' with the following code:

```
1 #include <iostream>
2 #include <thread>
3 #include <mutex>
4 #include <vector>
5 #include <queue>
6 #include <condition_variable>
7 #include <chrono>
8 #include <algorithm>
9 #include <string>
10
11 using namespace std;
12
13 class Elevator {
14 private:
15     int currentFloor;
```

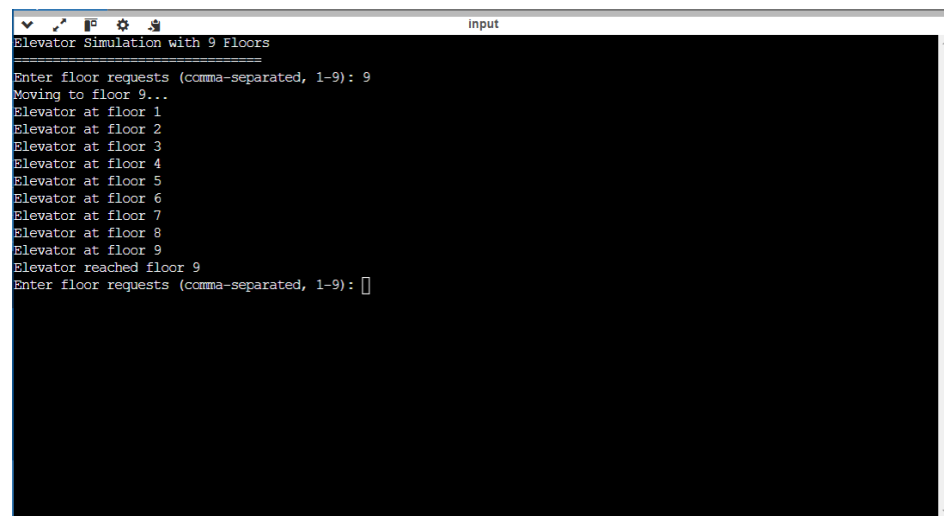
Below the code editor is a terminal window titled 'input' showing the program's output:

```
Elevator Simulation with 9 Floors
Enter floor requests (comma-separated, 1-9): []
```

B. Usage Instructions

Once you run the program you can now use the program.

- a. **Choosing a single floor:** when choosing a single floor just type in the number of the floor that you want to go to (only floors available 1-9).



The screenshot shows the terminal window from the previous image, now displaying the program's execution. The output is as follows:

```
Elevator Simulation with 9 Floors
Enter floor requests (comma-separated, 1-9): 9
Moving to floor 9...
Elevator at floor 1
Elevator at floor 2
Elevator at floor 3
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9): []
```

- b. **Choosing multiple floors :** when choosing multiple floors just type in the numbers of the floors that you want to go to, it should be separated by comma sign (,)(only floors available 1-9).

```
input
Elevator Simulation with 9 Floors
=====
Enter floor requests (comma-separated, 1-9): 2,3,6,9
Moving to floor 2...
Elevator at floor 1
Elevator at floor 2
Elevator reached floor 2
Moving to floor 3...
Elevator at floor 3
Elevator reached floor 3
Moving to floor 6...
Elevator at floor 4
Elevator at floor 5
Elevator at floor 6
Elevator reached floor 6
Moving to floor 9...
Elevator at floor 7
Elevator at floor 8
Elevator at floor 9
Elevator reached floor 9
Enter floor requests (comma-separated, 1-9): 
```

X . Challenges and Solutions

XI . Future Enhancements

XII .Conclusion

XIII .References

XIV. Appendices