ESE 224
Spring 2017

Final Project
Elevator Bank Design and Implementation

Score: 36 points

1 Project Description

The goal of this project is to design and develop C++ code and algorithms to control a bank of elevators, which services many floors and transports people "efficiently ", that is, as per the given specifications. A second goal is to effectively employ object oriented design, through appropriate use of classes, data structures and Standard Template Libraries (STLs).

1.1 Problem Specifications

- The elevator bank has m elevators and services n floors. As an example, you may choose $m = 3$ and $n = 10$.
- Each elevator can stop at every floor.
- The direction of an elevator has three states *up*, *down* and *standing*.
- Each elevator can carry up to a max of $Nmax$ persons.
- The program loops through discrete time steps $t = 1, 2, \ldots Tstop$, where $Tstop$ is an input parameter.
- At each time step, either 0, 1 or 2 persons arrive at a floor. This information can be read from an input data file, as described in 1.3.
- An elevator takes k time step to move up or down by k floors, if it doesn't stop. For example, to move from floor 4 to floor 5 requires one time step. And to move from floor 7 to floor 3 requires 4 time steps.
- When an elevator stops at a floor, for either passenger pick up or drop off, it does so for *one time step*.
- When a person arrives on a floor, they are **assigned** the "nearest "elevator. Here nearest is defined in the number of time steps. For example, let us say a person arrives on floor 5, and wants to go up. Elevator $E1$ on floor 2, moving in the *up* direction, with no stops, is 3 time steps away. Elevator $E2$ on floor 6, moving in the *down* direction and headed for floor 3 is 6 time steps away. Therefore, the person is assigned elevator $E1$.
- Each arriving person has an arrival floor, destination floor and the assigned elevator as its data members.
- When a person is assigned an elevator, their arrival floor and destination floor go in the *queue* or the *list* of the elevator stops.

- Each elevator contains a list of stops, which gets updated at each time step.
- An elevator moving up (down) continues to move up (down), until it exhausts all the stops in its list.

1.2 Class Design and STLs

You must make appropriate use of classes and class objects, class data members, class methods, constructors/destructors to implement your code. Here are some suggested classes and their data members:

- Person class with ID, arrival floor, destination floor and assigned elevator. An example ID for a person maybe $P1A3D5$, for person 1, arrival floor 3 and destination floor 5. This will make it easier to generate and track test input and output.
- Elevator class with direction, number of people and list of stops.

Think of the STL containers you will be utilizing. For example, a *list* allows for easy insertion and removal of elements, a *vector* can change size, and you can push and pop elements at the end of it, a *deque* allows for fast insertion and deletion of elements at both ends.

1.3 Input Data

The input data consists of the arrivals on each floor, at time steps $t = 1, 2, \ldots TStop$. At each time step, the program reads in each arriving person, for each floor, and processes this data. For example, for $n = 10$ floors, and at time step $t = 1$, the arrivals maybe $P1A1D4$ and $P2A1D5$ on floor 1, $P1A3D1$ and $P2A3D7$ on floor 3, and 0 on the remaining floors.

1.4 Program Output

Generate appropriate output for a given input test data, which displays the information, that is, people getting on and off each floor, for each time step, for say 10 consecutive time steps.

2 Teams and Interview

You may work alone or in teams of two. Interviews will be conducted for all teams, and each member of the team should be able to discuss their project and answer questions.

3 Project Report

The project *must* include a report which describes the program design, including class definitions, STL data structures and the algorithms. You may make use of figures to illustrate your design.

4 Assignment Submission

The Assignments **must** be submitted on the Blackboard, and should include the following:

1. Project Report in .pdf format.
2. Some test cases, with input and output data file.
3. All the C++ source code in .cpp format, and header files if any, in .h format.

4.1 Hardcopy

Please also submit a hard copy of the project, for ABET accreditation.