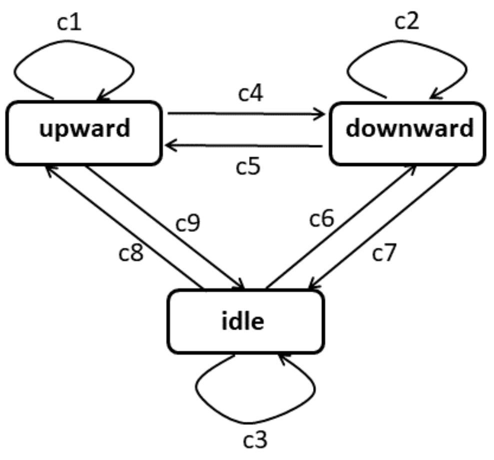
Assignment 6 – Part 3

(may be done by a team of at most two students)

Due: Thursday, December 6 (11:59 pm)

The behavior of an elevator controller is well-suited for State Chart specification. Consider a single elevator serving multiple floors. The elevator moves in one direction serving all requests in that direction before reversing its movement. For example, if the elevator is moving up, it goes to all higher-numbered floors chosen by people inside the elevator or by people outside the elevator who pressed the ‘up’ button. After serving all such requests, it continues moving upward to the highest floor where the ‘down’ button has been pressed. The processing of requests during the downward movement is similar, and the two movements are repeated indefinitely.

The picture below shows the state transitions for the elevator controller. It also shows three ordered lists which are automatically updated as new requests arrive and existing requests are served. Each transition label stands for a *guard condition* written in terms of three built-in predicates, `isempty()`, `min()`, and `max()` on ordered lists.



Ordered Lists (*ascending order*):

* **to** = list of floors to which people inside elevator want to reach in the direction of its motion
* **down** = list of floors outside the elevator from where people have pressed the ‘down’ button
* **up** = list of floors outside the elevator from where people have pressed the ‘up’ button

The definitions of the built-in predicate are as follows:

list.`min()` – return the smallest value in a non-empty ordered list of numbers;
list.`max()` – return the largest value in a non-empty ordered list of numbers;
list.`isempty()` – return true if the ordered list is empty; other return false of numbers.

Your task in this part of the assignment to write the guard conditions c1 .. c9. Sample guard conditions are shown below. In these conditions, the variable `f` stands for the elevator’s current floor. This variable is automatically updated as the elevator moves from one floor to another.

```
c1 = !to.isempty()  || (!up.isempty() && up.max() ≥ f)  ||
                       (!down.isempty() && down.max() > f)

c4 = to.isempty()  &&  ((!up.isempty() && up.max() < f)  ||
                        (!down.isempty() &&  down.max() ≤ f))

c7 = to.isempty()  &&  up.isempty()  &&  down.isempty()
```

Prepare a file `StateChart.pdf` containing the state chart picture and the definitions for all nine guard conditions c1..c9.

Note: In general, the label on the transition of a UML state chart has the form *event / guard / action*. For the elevator problem, an *event* corresponds to pressing a button and an *action* corresponds to updating one of the ordered lists or the floor number. For simplicity, you are not asked to consider events and actions in this assignment; it suffices to show the guard conditions for the transitions. You are also ***not*** required to use Star UML for this part of the assignment.

***What to Submit*.** Prepare a directory named *A6_Part3_UBITId1_UBITId2* if this part is done by a team of two students; otherwise, name it as *A6_Part3_UBITId* if done solo. (Order the *UBITId*s in alphabetic order, in the former case.) In this directory, place the file `StateChart.pdf`. Compress the directory and submit it using `submit_cse522`. Only one submission per team is required.

End of Assignment 6 – Part 3