C SC 230 – Summer 2014 – Assignment 3
Elevator Simulation: Design and Test

1. Developing the Program

Unless you are a super programmer and don't need any advice, you *should* follow the stages of development shown in Table 1. This is the *only way* to ensure that you get partial credit for a program which is not perfect. The table decomposes the problem into a sequence of tasks. After completing any stage, you will have a working program. It won't perform all the actions required of the final program but it will perform some of them. Moreover it is strongly suggested that you think in even smaller steps by dividing up each stage into components and test each one separately. The testing sequence which is to be used at the demo session is also available below and its steps are linked to the stages of Table 1.

As an example, suppose you need to develop the code for capturing whether one of the two black buttons have been pushed by the user and then you need to attach an appropriate action to each. Consider implementing using the following small steps:

(a) Write the code to wait for the user to press any black button. When a button is pressed, print a message to `stdout` to indicate the event (copy the code from examples in labs).
(b) Once the above is working correctly, add the code to distinguish between a left black button or a right black button. Again, simply print a message to `stdout` for testing.
(c) Call the appropriate function for the action attached to the button push. Simply print a message again stating that you entered the function correctly.
(d) Now you are ready to start implementing the action itself in the function. Test the action and delete the temporary messages to `stdout`.

By implementing very small portion of code and testing it every time with a simple message, you will know that the logic is proceeding correctly before it gets more complex, and you can manage, with breakpoints in strategic places, to trace the execution.

IMPORTANT! You will not get any credit for stage k+1 unless you score at least 80% of the marks available for stages 1 through k. Make sure you properly test your program before starting on the next stage.

Table 1: Program Development Stages

Stage	Program Tasks to Implement
0	The program consists of a main function which calls the `Initdraw` function to initialize all displays and then calls the `Idling` function. The elevator idles at floor 1 while waiting for inputs. The simulated time is updated every 1 second. To exit the program, the Stop key is pressed and the outputs are updated. *This part, including some extra functions for updating variables and calculating data is given to you in the template file. Make sure you understand it all and test it.*
1	**Goal:** Starting at floor 1, the elevator moves up one floor after the blue buttons C2 is pressed. The floor number is updated and the elevator idles at floor 2. **Code:** In `Main`, the "SwitchOnBlue" cases are listed with appropriate branches to individual code segments (still in `Main`). Only the code segment for the detection of C2 is implemented with the objective of moving upwards. Thus implement the function "`MovingUp`", on return of which you go back to `Main` and then to `Idling`. Implicitly you need to implement "`WaitAndPoll`". Why? Because as you are moving from floor 1 to floor 2 for 5 seconds you need to keep polling for any other button press. Note that you need to implement the *polling* portion, not necessarily yet the action portion. That is, poll in "`WaitAndPoll`" and return in R0 a correct value, but then do not bother yet to check it and attach any action to it yet. Make sure you do `MovingUp` first using just a simple delay without polling, and subsequently introduce `WaitAndPoll` (2 sub-stages!).

Table 1: Program Development Stages (Continued)

Stage	Program Tasks to Implement
2	**Goal:** Starting at floor 1, the elevator moves up after one of the blue buttons C2, C3 or C4 is pressed. The floor number is updated when passing by a floor and the elevator returns to idling at the final floor. **Code:** An extension of above. You need to implement two more cases in the "SwitchOn-Blue"construct and use "`MovingUp`" appropriately. **Check:** Go see your manager and have your code looked at. This step involves a lot of new items and it is best to know right away if you are on the right track with the code structure.
3	**Goal:** Starting at floor 1, the elevator moves up after one of the blue buttons C2, C3 or C4 is pressed. The floor number is updated when the elevator passes by any new floor. When reaching the destination floor, the doors open and the output is changed accordingly. Then the elevator goes back to idling at that floor. **Code:** Add in "`MovingUp`" the code for checking whether the destination floor has been reached. If yes, doors must open and the code for opening the doors is needed now, displaying it on the screen and pausing; this is done by designing the "`OpenDoors`" function. Note that while the doors are opening, you wait while polling. Again, poll but do not attach any further action yet to the result. (Filename=`Elevator3.s`)
Testing Scheme: 1-3	
4	**Goal:** Starting at floor 1, the elevator moves up after the blue buttons C4 is pressed. The floor number is updated when the elevator passes by each new floor. In the meantime, as soon as it starts moving, the buttons for FU2 and FU3 are pressed. The elevator stops at floors 2 and 3 and opens the doors, then continues to floor 4 and opens the doors. Then the elevator idles at floor. 4 **Code:** design the code for: (a) Setting the global arrays correctly after capturing the FU2 and FU3 button presses; this is done by designing the "`SetButtonsArray`" function which is called by "`WaitAndPoll`". (b) When arriving at a new floor, checking the global array to see whether you should stop and open the doors or not; this is done by designing the "`CheckFloorDoors`" function; (c) Write the "`CheckSignalsHigher`" function which checks the global array, while moving up, to see whether there are any signals for any floor about the current one being passed. If the answer is yes, then continue moving up, else idle at current floor. (d) When opening the doors at a floor, you should clear the global array of the signal at that floor (so you do not stop again); this is done by designing the "`ClearButtonsArray`" function and calling it from "`OpenDoors`". (Some of these functions may already been given to you).
5	**Goal:** At any point in time when polling, the black left button for stopping the simulation should also be considered. During any of the simulations above, if the left button is pressed, the program should terminate gracefully (as in the specs). This implies that, no matter when and where the Stop button is detected, the software control returns all the way to `Main`, from which the only exit point from the program should be. No exiting from any other point! **Code:** Make sure that after "`WaitAndPoll`" returns, a check is made on its result to terminate every time, no matter who called it.

Table 1: Program Development Stages (Continued)

Stage	Program Tasks to Implement
6	**Goal:** Starting at floor 1, the elevator moves up after the blue buttons C4 is pressed. The floor number is updated when the elevator passes by each new floor. In the meantime any of the buttons FD2 or FU2 and FU3 or FD3 are pressed. The elevator stops at floors 2 and 3 and opens the doors, then continues to floor 4 and opens the doors. Then the elevator goes back to idling at that floor. **Code:** You need to implement more cases in the "SwitchOnBlue"construct. (File-name=`Elevator4.s`)
Testing Scheme: 4-6	
7	**Goal 1:** The same phases as above are implemented for moving down and they are integrated with moving up. The code is similar. Move the elevator to floor 4, then press C1. The elevator moves down, the floor numbers are updated and the elevator opens the doors at floor 1. **Goal 2:** Move the elevator to floor 4, then press C3. While the elevator moves down, one or more of the buttons at a lower floor are pressed. The elevator stops at floors 3 and opens the doors, then continues to one or more of the requested lower floors and opens the doors. Then the elevator goes back to idling at the final floor. **Code:** The extra code for moving down is a mirror image of the code so far for moving upwards. Do not start this code until you have fully tested going up and until you have checked with your manager that you are on the right track.
Testing Scheme: 7-8	
8	**Goal:** Any button press should also be captured while the doors are opening and the elevator is stopped at a floor. **Code:** Make sure that "`WaitAndPoll`" is called properly and its result acted upon also when opening doors.
Testing Scheme: 9	
9	**Goal:** Whenever the Emergency button is pressed, the program exits to the emergency state with the corresponding flashing displays. The only way to stop execution is to use the Stop button in ARMSim#. This shows you that it is possible to exit a program from a place other than `Main`, but it should be reserved only for super emergencies and unusual situation (not ever for normal exceptions). **Code:** Make sure that "`WaitAndPoll`" is called properly and the emergency result should be tested and acted upon first and immediately.
Testing Scheme: 10	
10	**Goal:** The program works on the Embest Board.
Total = 30 marks for correct execution (from above) + 20 marks for code evaluation (documentation, structure, correct use of subroutines, adherence to specifications, design, etc.)	

2. Demo Test Sequence

Each bold line separator in the table indicates the start of a new stage in the program capability. No credit is given for stage *k*+1 unless stage *k* was fully implemented, i.e., testing stops at a solid bar if the previous tests are not working reasonably well. ('Reasonably well' = about 80% of marks awarded.) Please note that changes may be made to this testing sequence.

#	Action	Things to Observe	Mark	Tot
0	Start up the program (if failure, stop and grade = 0)	• Initial screen is displayed. Elevator idles at floor 1 (patterns).	0	0
1	C2 is pressed. (Observe going up and opening doors at final destination.)	• UP pattern, left LEDs on, for X seconds • Then floor number = 2 on screen and 8-segment display • OPEN DOORS pattern, both LEDs on, for Y seconds • Elevator idles at floor 2 (patterns).	2	
2	Press C3. (Observe it going up again and opening doors at final destination.)	• UP pattern, left LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • Elevator idles at floor 3 (patterns).	2	
3	Press left black button.	• Exit from elevator simulation program	1	
4	Reload the program. Press C4. (Observe changing floor numbers as it moves up and opening doors at final destination.)	• UP pattern, left LEDs on, for X seconds • Floor number =2 on screen and 8-segment display. • UP pattern, left LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • UP pattern, left LEDs on, for X seconds • Floor number =4 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y second • Elevator idles at floor 4 (patterns).	2	7
If there are no OPEN DOORS				-2
If no nested signals, test above plus some moving down in sequence (going to Idling in between)				-3
5	Reload the program. Press C4. As soon as it starts moving up, press FU3. (Observe capturing extra signals and stopping at the designated floors.) See also Note below.	• UP pattern, left LEDs on, for X seconds • Floor number =2 on screen and 8-segment display. • UP pattern, left LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y second • UP pattern, left LEDs on, for X seconds • Floor number =4 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y second • Elevator idles at floor 4 (patterns).	3	
6	Reload the program. Press C4. As soon as it starts moving up, press FD2 and FD3. (Observe capturing extra signals and stopping at the designated floors.) See also Note below.	• UP pattern, right LEDs on, for X seconds • Floor number =2 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • UP pattern, left LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • UP pattern, left LEDs on, for X seconds • Floor number =4 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • Elevator idles at floor 4 (patterns).	3	13

#	Action	Things to Observe	Mark	Tot
7	Press C1. As soon as it starts moving down, press FD3 and FU2. (Observe capturing extra signals and stopping at the designated floors.) See also Note 2.	• DOWN pattern, right LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • DOWN pattern, left LEDs on, for X seconds • Floor number =2 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • DOWN pattern, left LEDs on, for X seconds • Floor number =1 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • Elevator idles at floor 1 (patterns).	4	
8	Press C2. As soon as it starts moving up, press FD4. (Observe capturing extra signals and stopping at the designated floors.) See also Note 2.	• UP pattern, left LEDs on, for X seconds • Floor number =2 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • UP pattern, left LEDs on, for X seconds • Floor number =3 on screen and 8-segment display. • UP pattern, left LEDs on, for X seconds • Floor number =4 on screen and 8-segment display. • OPEN DOORS pattern, both LEDs on, for Y seconds • Elevator idles at floor 4 (patterns).	4	21
9	Press C1. As soon as it starts moving down, press C3. *While doors are open at floor 3*, press FD2 and FD4.	• DOWN pattern, right LEDs on, from 4 to 3. • OPEN DOORS pattern, both LEDs on at floor 3 • DOWN pattern, right LEDs on, from 3 to 2. • OPEN DOORS pattern, both LEDs on at floor 2. • UP pattern, left LED on, from 2 to 4. • OPEN DOORS pattern, both LEDs on at floor 4 • Elevator idles at floor 4 (patterns).	2	23
10	Choose a couple of the above sequences again and at some point press the right black button (Emergency).	The program exits to the Emergency state - to quit, one must use the Stop button of ARMSim#. • All segments flashing on/off every 0.5 seconds on 8-segment display. • Both LEDs flash on/off together every 0.5 seconds. • Emergency message only on screen.	3	26
11	The above tests appear to all work on the Embest board. Student must show that they can build a project and download it to the board. (Full testing is unnecessary.)		4	30
Total = 30 marks for correct execution (from above) as tested by TAs + 20 marks for code evaluation (documentation, structure, correct use of subroutines, adherence to specifications, design, etc.), as evaluated by instructor (in one-on-one meeting). Max 14/30 marks if no nested signals.				

Note. The simpler algorithm requested here assumes that the elevator stops at a floor when *any* signal is present related to that floor. Consider the following scenario: moving up from floor 1 to floor 3, there is a request at floor 2 with the FD2 button, that is, a request to go down from floor 2. The current specification have the elevator stop at floor 2 to collect a user even if the elevator is moving up and the user wants to go down. It is assumed that the user will take an extra ride up to floor 3 and will eventually push another button inside the elevator car to go down. The much better implementation would differentiate between up and down signals request. Thus, in the case above, the elevator would not stop at floor 2 while going up, but would detect the FD2 request and them move down to floor 2 after having gone to floor 3. If anybody states that they have implemented this extra algorithmic step (correctly), test for it and be prepared to give +3 extra marks as bonus.