Assignment Part2 (Promela and Spin).
Due date 29 April, 2022

You have to upload your solution by the due date via Google classroom. You should submit a zip containing the Promela source files and an additional document with the required explanations.

1. Consider the design of an elevator controller for a 4 story building (floor 0, 1, 2 and 3), described below. We would like to model this controller in Promela and analyze some safety features of it using the Spin model-checker. A partial Promela model for this design is given on the next page.

   In our design, the elevator has four main functionalities which can be executed by sending it the following messages: `open`, `close`, `up` and `down`. The elevator's doors are opened when the message `open` is received, and closed when the message `close` is received. The elevator can be commanded to move one floor up by using the message `up` and one floor down by using the message `down`.

   On each floor of the building there is a button the user can press in order to call the elevator (to that floor). We model the user's elevator call by four messages that are sent to the elevator's controller: `call_0`, `call_1`, `call_2`, and `call_3`, where `call_i` expresses calling the elevator to floor `i`.

   Inside the elevator there are also four buttons a user can press and which will take her to the selected floor. We model the user's elevator command by four messages which again are sent to the controller: `go_0`, `go_1`, `go_2`, and `go_3`, where `go_i` expresses taking the user to floor `i`.

   The elevator controller's aim is to "read" the user's choices (calling the elevator to a floor, taking her to a floor) and instruct the elevator about the procedure to follow in establishing the selected choice.

   In the initial state, the elevator is at floor 0 and its doors are closed.

```
/* Partial Promela model of an elevator. */

mtype = { call_0, call_1, call_2, call_3,
          go_0, go_1, go_2, go_3,
          open, close,
          up, down}

chan floor_buttons = [1] of { mtype };
chan elevator_buttons = [1] of { mtype };
chan commands = [0] of { mtype };

active proctype elevator() {
    do
        :: commands ? open -> printf("Elevator: opened doors.\n");
        :: commands ? close -> printf("Elevator: closed doors.\n");
        :: commands ? up -> printf("Elevator: moved up one floor.\n");
        :: commands ? down -> printf("Elevator: moved down one floor.\n");
    od
}

/* Simulates random pushing of call buttons. */
active proctype floor_button_input() {
    do
        :: floor_buttons ! call_0;
        :: floor_buttons ! call_1;
        :: floor_buttons ! call_2;
        :: floor_buttons ! call_3;
    od
}

/* Simulates random pushing of elevator buttons. */
active proctype elevator_button_input() {
    do
        :: elevator_buttons ! go_0;
        :: elevator_buttons ! go_1;
        :: elevator_buttons ! go_2;
        :: elevator_buttons ! go_3;
    od
}

active proctype controller() {
    int at = 0;
    bool closed = true;

    /* Implement your own elevator controller here! */

}
```

a) Model the elevator's controller in the `controller proctype`. Explain your modeling choices (e.g., the used Promela statements, sequence of statements to be executed based on the user choice, etc.).

b) Add assertions to the Promela model that trigger if the elevator receives the `up` message while at floor 3 or if the elevator receives the `down` message while at floor 0. (Please include the full Promela model in the submitted solution.) Check with Spin whether the assertions are triggered. Explain the obtained result. If the assertions are triggered, provide a corrected model. (Please include a Spin run log with your explanations.)

c) Add assertions to the Promela model that trigger if the elevator receives the `up` or `down` messages while its doors are open. (Please include the full Promela model in the submitted solution.) Check with Spin whether the assertions are triggered. Explain the obtained result. If the assertions are triggered, provide a corrected model. (Please include a Spin run log with your explanations.)

d) Mention and explain some properties that should be satisfied by the system/controller that you have designed (that can be expressed using LTL). Check with Spin whether the LTL properties are satisfied by the model (explain the obtained result).