CSCE4930-01

Multi-FSM simulator Report

Mohamed Ashraf Taha

ID: 900172754

Submitted to:

Dr. Sherif El-Kassas

Eng. Mahmoud Esmat

5/12/2021

I. Aim

This report aims to explain the changes made to the previous assignment of the FSM simulator. This assignment I introduced the concept of multi-threading and mutexes to the assignment.

II. Changes to FSM simulator

A. In the FSM simulator I was terminating the FSM by doing exit(0) in the endAction.cpp runAction. However, I changed that since it terminates the whole program and not only the FSM instance running. I changed that by checking in the runFSM method by checking if the ActionType is end we return from the function to the main TestingFunctionality.cpp .

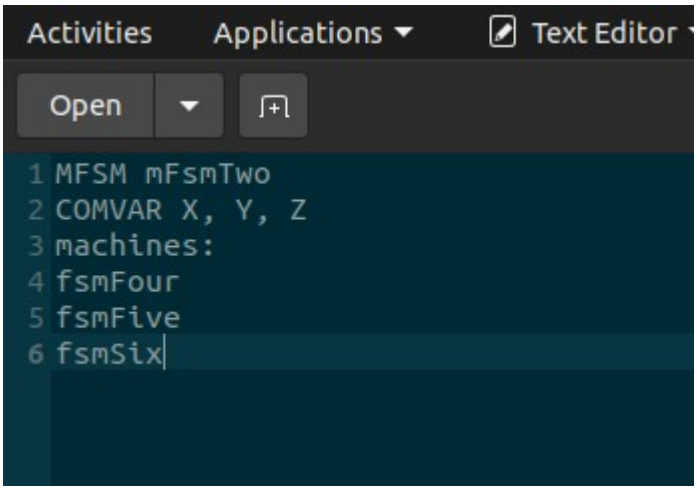
B. Parsed the mfsm file and extracted the common variables and the machines names, used functions that I used before in the FSM assigment, so MFSM inherits from FSM. Then I stored the common variables in a vector of type VAR, which is a class that I already implemented from the last assignment. Then I passed the vector of common variables by reference (&) to the FSM and from the FSM I passed to parse states method then from parse states to the following three actions: add; responsible for handling the expressions, wait; responsible for waiting for user input from the screen and out; responsible for outputting to the screen values or string,

C. For the mutexes part, my code mainly depends on two mutexes, one for the add “Expression” action and one for **both** the wait and the out actions.

Then I lock before the expression. I also lock, using the same mutex the wait and the out actions since we dont want a thread to out something while the user is inputting.

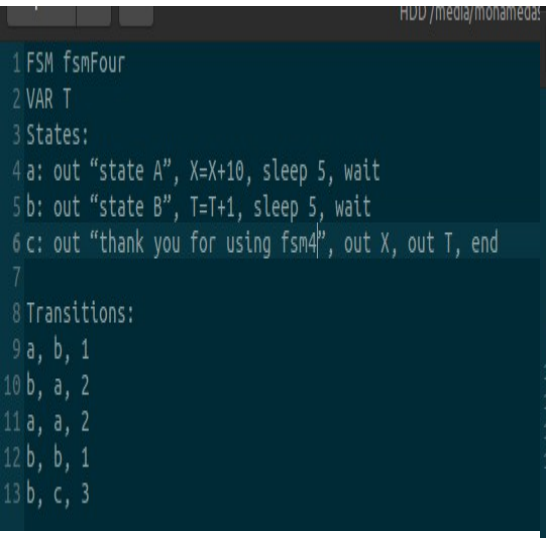
III. Debug of complex example

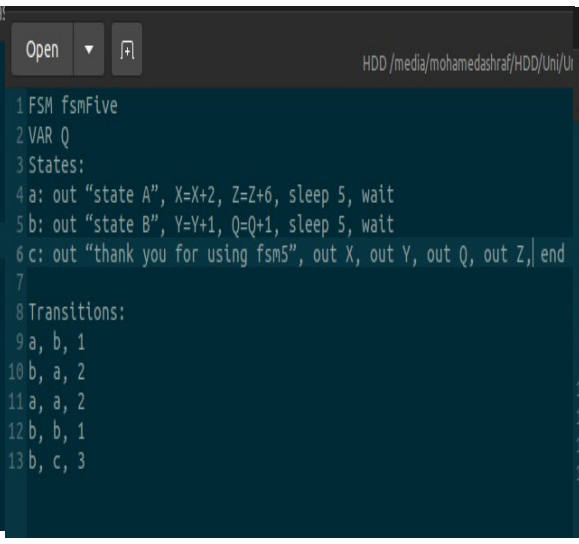
mfsmTwo:

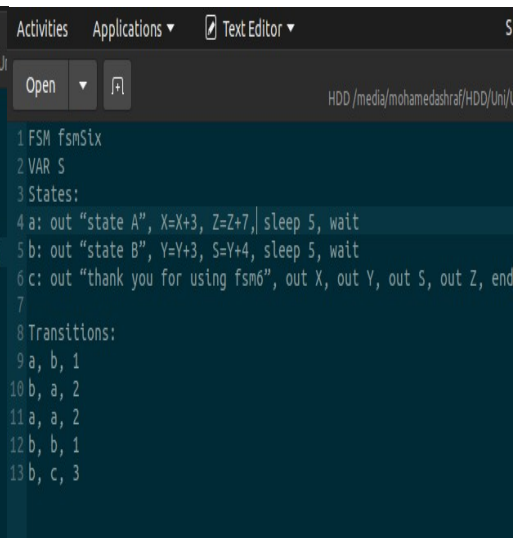


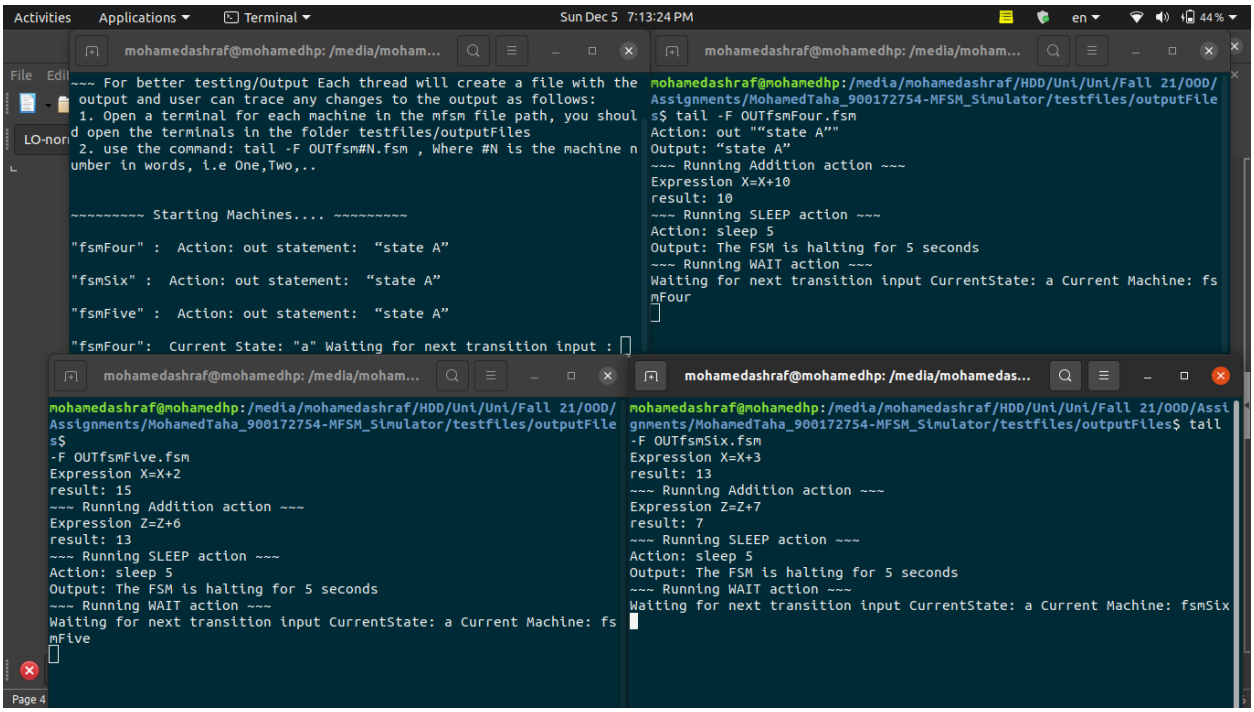
the three machines:

FsmFour fsmFive FsmSix









as we can see here:

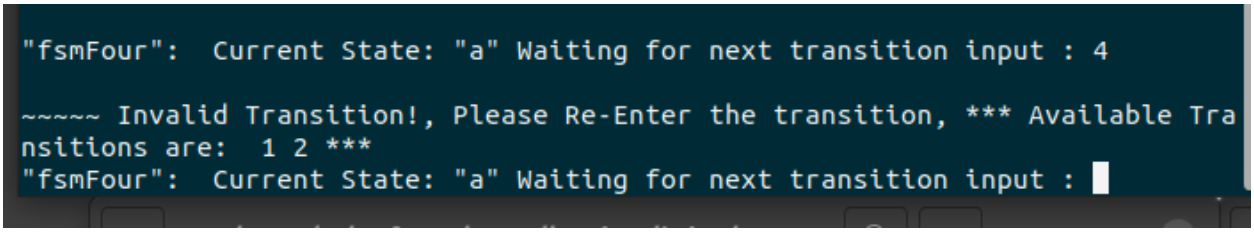
- machine four started first so value of x became 10

- then machine six next so value of x became 13

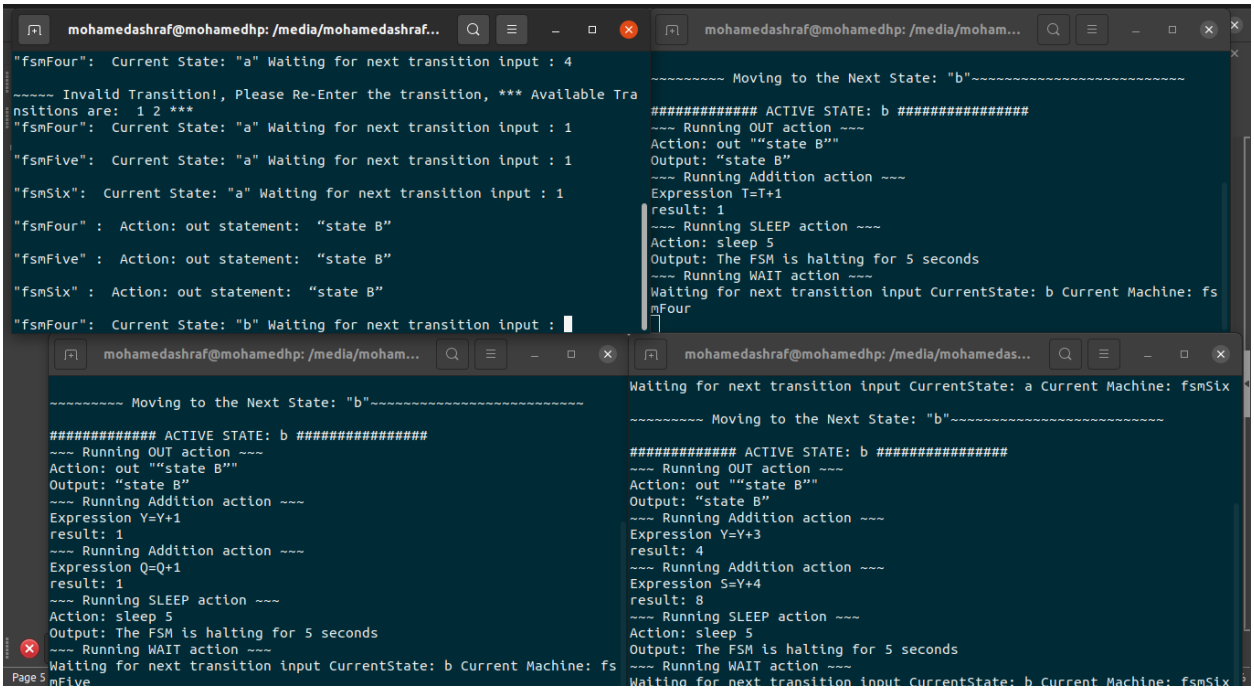
- then machine five next so value of x is now 15

- since six started first so value of Z became 7, then five started and added z=z+6 -> 7+6 =13

- Fault tolerance part

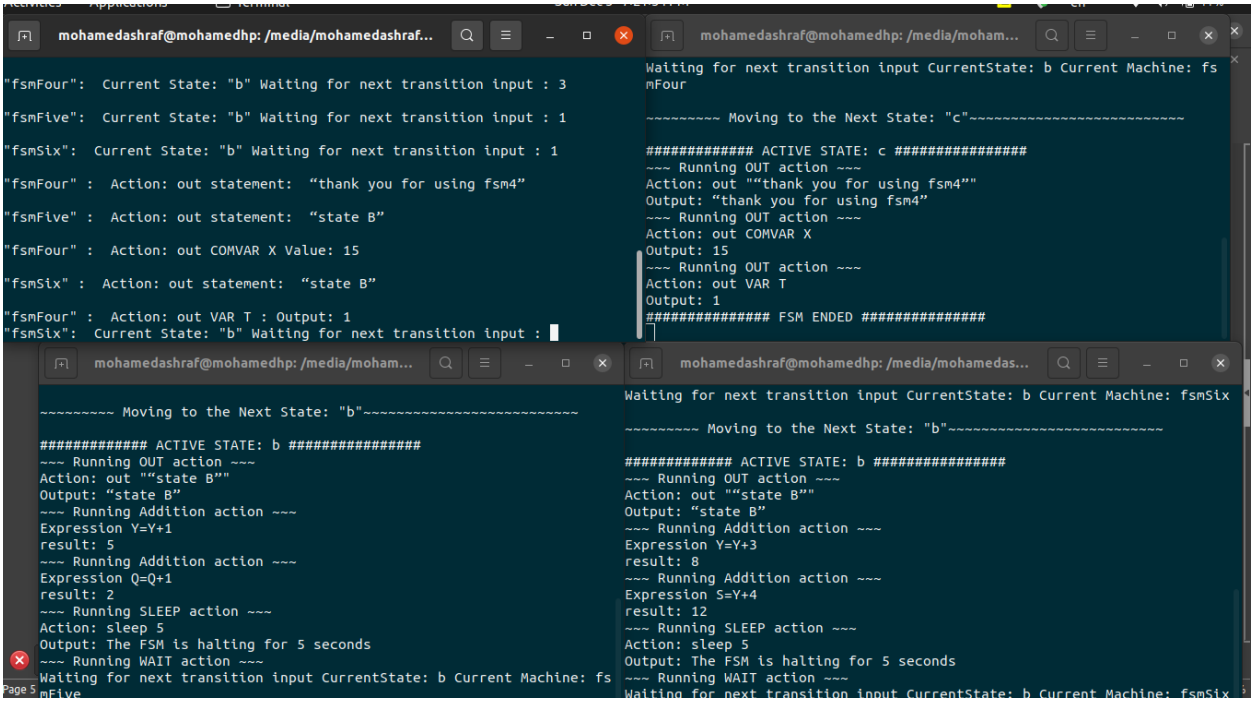


- then we moved to state B



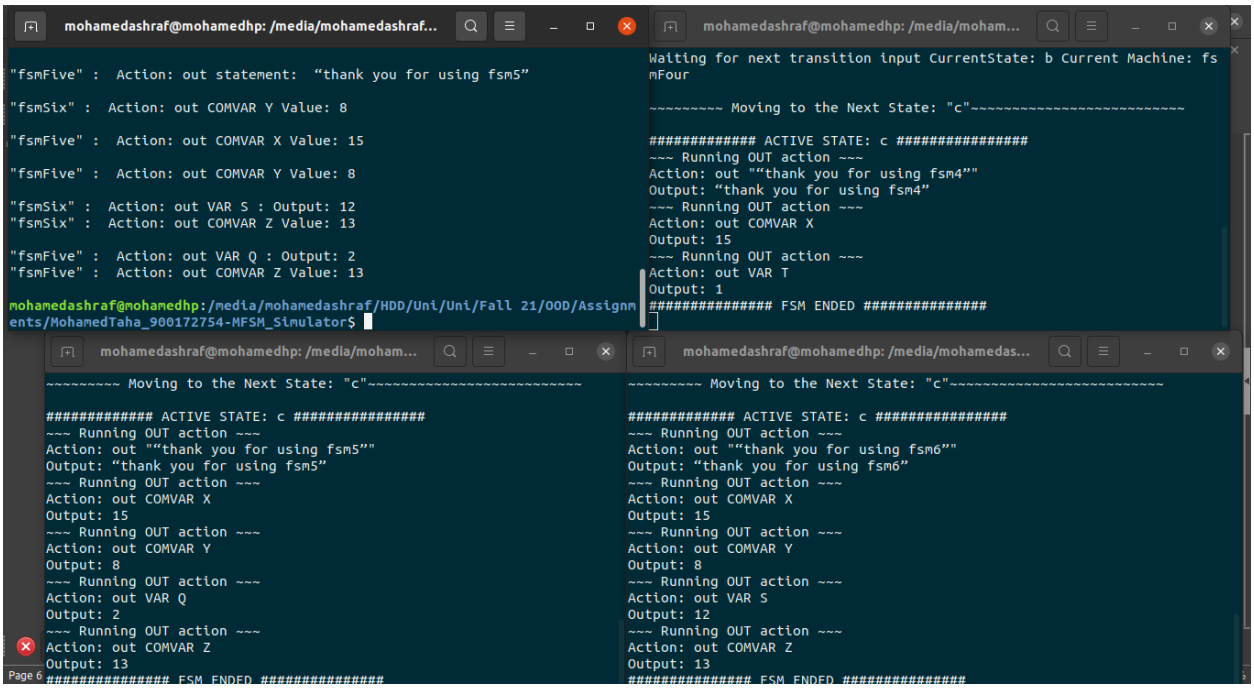
- T (Local variable) became 1 in machine four

- Then Y(global variable became) 1 in machine five, then six ran next and Y became = 1+3 -> 4, then S= Y+4, and Y is now 4 then S = 8



-Now I ended the fourth machine then it outputted the value of X it holds now, which is 15 and the value of T as well which is 1.

- then machines five and six moved to state B again , so Y became 5 in machine five and then in machine six we made 5+3 , so Y is now 8 and accordingly S is 12, we then increased Q to 2.



- FSMseven -> Has no variables but has the Keyword VAR

-FSMeight -> Has no VAR section