S1 (3 hours and 15 min work time)

1. **(0.5p by default). Please extend your Java-based interpreter with the following features:**
   a. (0.1p). Define a unique identifier (of type integer) for each program state.
   b. (0.8p). Define a new global table (global means it is similar to Heap and Out tables and it is shared among different threads), LockTable that maps integer to integer. It has a fixed size of 3 and it is initialized to 0. LockTable must be supported by all previous statements.
   c . (0.8p). Define the new statement
   lockEnter(number)
   where number is an index from the LockTable. Its execution on the ExeStack is the following:
   - pop the statement
   - if number is not an index in LockTable return an exception and terminate the execution
   - elseif LockTable[number]==0 then LockTable[number]=prgstate id else push back the lock statement

   d. (0.8p) Define the new statement:
   lockExit(number)
   where number is an index from the LockTable. Its execution on the ExeStack is the following:
   - pop the statement
   - if number is not an index in LockTable then do nothing
   - if LockTable[number]== id of the current prgstate then LockTable[number]=0 else do nothing

   e. (1.25p) Extend your GUI to suport new statements both the program input and step-by-step execution. **If you have only a text interface you can get maximum 0.25p for this feature.**
   f. (0.75p). Show the step-by-step execution of the following program. At each step display the content of each program state (all the structures of the program state). The step-by-step execution must be displayed on the screen and also must be saved into a text readable log file.
   v=10;lockEnter(1);print(v);fork(v=20;lockEnter(1);print(v);lockExit(1));print(v+1);lockExit(1)

2. **(0.5p by default) Please extend your C#-based interpreter with the following features:**
   a. (0.1p). Define a unique identifier (of type integer) for each program state.
   b. (1.2p). Define the new statement which writes in the heap:
   wh(varname,exp)
   Its execution on the ExeStack is the following:
   - pop the statement
   - get from the SymTable the address associated to the variable varname
   - write at that address in the Heap the result of the expression exp evaluation

   c. (1.2p). Define the new expression which reads from the keyboard:
   read()
   Its evaluation calls a C# method which reads an integer from the keyboard. C# method may print a message like "Introduces an integer for ToyLanguage " and returns that integer.
   d. (1.25p). Extend your GUI to suport new statements and expressions both the program input and step-by-step execution. **If you have only a text interface you can get maximum 0.25p for this feature.**
   e. (0.75p). Show the step-by-step execution of the following program. At each step display

the content of each program state (all the structures of the program state). The step-by-step execution must be displayed on the screen and also must be saved into a text readable log file.

```
v=new(read());print(r(v));fork(wh(v,1+read());print(r(v));fork(wh(v,20));print(r(v)));print(r(v)+1)
```