Practical Subjects – 16 January 2020

Work Time: 2 hours and 30minutes
Please implement in Java the following two problems.
If a problem implementation does not compile or does not run you will get 0 points for that problem (that means no default points)!!!
If for one problem you have only a text interface to display the program execution you are penalized with 1.25 points for that problem.

1. **(0.5p by default). Problem 1:Implement a ToySemaphore mechanism in ToyLanguage.**
   a. (0.5p). Inside PrgState, define a new global table (global means it is similar to Heap, FileTable and Out tables and it is shared among different threads), SemaphoreTable that maps an integer to a tuple: an integer,a list of integers and another integer. SemaphoreTable must be supported by all of the previous statements. It must be implemented in the same manner as Heap, namely an interface and a class which implements the interface.
   b. (0.75p). Define a new statement
   newSemaphore(var,exp1,exp2)
   which creates a new semaphore into the SemaphoreTable. The statement execution rule is as follows:
   Stack1={newSemaphore(var, exp1,exp2)| Stmt2|...}
   SymTable1
   Out1
   Heap1
   FileTable1
   SemaphoreTable1
   ==>
   Stack2={Stmt2|...}
   Out2=Out1
   Heap2=Heap1
   FileTable2=FileTable1
   - evaluate the expression exp1 and exp2 using SymTable1 and Heap1 and let be number1 and number2 the results of this evaluation
   SemaphoreTable2 = SemaphoreTable1 synchronizedUnion {newfreelocation ->(number1,empty list,number2)}
   if var exists in SymTable1 and has the type int *then*
   SymTable2 = update(SymTable1,var, newfreelocation)
   else throws an error
   Note that you must use the lock mechanisms of the host language Java over the SemaphoreTable in order to add a new semaphore to the table.
   c. (0.75p). Define the new statement
   acquire(var)
   where var represents an int variable from SymTable which is the key for an entry into the SemaphoreTable. Its execution on the ExeStack is the following:

- pop the statement
- foundIndex=lookup(SymTable,var). If var is not in SymTable or has not the type int then print an error message and terminate the execution.
- *if* foundIndex is not an index in the SemaphoreTable *then*
print an error message and terminate the execution
else
- retrieve the entry for that foundIndex, as SemaphoreTable[foundIndex]== (N1,List1,N2)
- compute the length of that list List1 as NL=length(List1)
- *if* ((N1-N2)>NL) *then*
if(the identifier of the current PrgState is in List1) *then*
- do nothing
else
- add the id of the current PrgState to List1

else
- push back acquire(var) on the ExeStack

Note that the lookup and the update of the SemaphoreTable must be an atomic operation, that means they cannot be interrupted by the execution of the other PrgStates. Therefore you must use the lock mechanisms of the host language Java over the SemaphoreTable in order to read and write the values of the SemaphoreTable entrances .

d**. (0.5p).** Define the new statement
release(var)
where var represents an int variable from SymTable which is the key for an entry into the SemaphoreTable. Its execution on the ExeStack is the following:
- pop the statement
- foundIndex=lookup(SymTable,var). If var is not in SymTable or has not int type then print an error message and terminate the execution.
- *if* foundIndex is not an index in the SemaphoreTable *then*
print an error message and terminate the execution
else
- retrieve the entry for that foundIndex, as SemaphoreTable[foundIndex]== (N1,List1,N2)
- *if*(the identifier of the current PrgState is in List1) *then*
- remove the identifier of the current PrgState from List1
else
- do nothing

Note that the lookup and the update of the SemaphoreTable must be an atomic operation, that means they cannot be interrupted by the execution of the other PrgStates. Therefore you must use the lock mechanisms of the host language Java over the SemaphoreTable in order to read and write the values of the SemaphoreTable entrances .

e**.(0.25).** Fix the problem of the unicity of ProgramState identifier. Each ProgramState must have an identifier that is unique. Note that this step perhaps

you already done at the laboratory. To be sure please also check whether v=1;fork(v=2);fork(v=3) generates ProgramStates with different identifiers.
f. (1p). Extend your GUI to suport step-by-step execution of the new added features. To represent the SemaphoreTable please use a TableView with three columns: an index, a value and a list of values.
g. (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 readable log text file. **The typechecker must not be adapted and called for this problem.**
The following program must be hard coded in your implementation.
Ref int v1; int cnt;
new(v1,2);newSemaphore(cnt,rH(v1),1);
fork(acquire(cnt);wh(v1,rh(v1)*10));print(rh(v1));release(cnt));
fork(acquire(cnt);wh(v1,rh(v1)*10));wh(v1,rh(v1)*2));print(rh(v1));release(cnt));
acquire(cnt);
print(rh(v1)-1);
release(cnt)
The final Out should be {20,200,199} or {20,19,200}.

2. **(0.5p by default) Problem 2: Implement Conditional Assignment statement in Toy Language.**
   a. (2.75p). Define the new statement:
   v=(exp1)?exp2:exp3

   Its execution on the ExeStack is the following:
   - pop the statement
   - create the following statement:
     if (exp1) then v=exp2 else v=exp3
   - push the new statement on the stack

   b. (1.75p). 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 readable log text file. **The typechecker must not be adapted and called for this problem.**
   The following program must be hard coded in your implementation:
   bool b;
   int c;
   b=true;
   c=b?100:200;
   print(c);
   c= (false)?100:200;
   print(c);

The final Out should be {100,200}