Practical Subjects – 22 January 2020

Work Time: 2 hours 45 minutes
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)!!!

1. **(0.5p by default). Problem 1: Implement Switch statement in Toy Language.**
   a. (2.75p). Define the new statement:
   switch(exp) (case exp1: stmt1) (case exp2: stmt2) (default: stmt3)

   It is a switch statement that executes either the statement stmt1 when exp==exp1, or the statement stmt2 when exp==exp2 or the statement stmt3 otherwise.
   Its execution on the ExeStack is the following:
   - pop the statement
   - create the following statement:
   if(exp==exp1) then stmt1 else (if (exp==exp2) then stmt2 else stmt3)
   - push the new statement on the stack

   The typecheck method of switch statement verifies if exp, exp1 and exp2 have the same type and also typecheck the statements stmt1, stmt2 and stmt3.
   b. (with a working GUI 1.75p, with a working text UI 0.5p). 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 following program must be hard coded in your implementation:

```
int a; int b; int c;
a=1;b=2;c=5;
(switch(a*10)
  (case (b*c) : print(a);print(b))
  (case (10) : print(100);print(200))
  (default : print(300)));
print(300)
```

   The final Out should be {1,2,300}

2. **(0.5p by default). Problem 2:Implement a CountSemaphore 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 pair: an integer, and a list of integers. 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. *Note that the lookup and the update of the SemaphoreTable must be atomic operations, 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 .*

b. (0.5p). Define a new statement

createSemaphore(var, exp1)

which creates a new semaphore into the SemaphoreTable. The statement execution rule is as follows:

Stack1={createSemaphore(var, exp1)| Stmt2|...}
SymTable1
Out1
Heap1
FileTable1
SemaphoreTable1
==>
Stack2={Stmt2|...}
Out2=Out1
Heap2=Heap1
FileTable2=FileTable1

- evaluate the expression exp1 using SymTable1 and Heap1 and let be number1 the result of this evaluation. If number1 is not an integer then print an error and stop the execution.

SemaphoreTable2 = SemaphoreTable1 synchronizedUnion {newfreelocation ->(number1,empty list)}

if var exists in SymTable1 and has the type int *then*

SymTable2 = update(SymTable1,var, newfreelocation)

else print an error and stop the execution.

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)
  - compute the length of that list List1 as NL=length(List1)
  - *if* (N1>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

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)
  - *if*(the identifier of the current PrgState is in List1) *then*
    - remove the identifier of the current PrgState from List1
  else
    - do nothing

e**.(0.5).** Implement the method typecheck for the statement createSemaphore(var, exp1) to verify if both var and exp1 have the type int. Implement the method typecheck for the statement acquire(var) to verify if var has the type int. Implement the method typecheck for the statement release(var) to verify if var has the type int.

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. (with a working GUI 0.75p, with a working text UI 0.5p). 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 following program must be hard coded in your implementation.

```
Ref int v1; int cnt;
new(v1,1);createSemaphore(cnt,rH(v1));
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 {10,200,9} or {10,9,200}.