Practical Subjects – 24 February 2017

Work Time: 3 hours
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 CyclicBarrier 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), BarrierTable that maps an integer to a pair of two integers. First element of the pair represents the list of threads identifiers waiting on the barrier, while the second value of the pair is the barrier value. BarrierTable 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
   newBarrier(var,exp)
   which creates a new barrier into the BarrierTable. The statement execution rule is as follows:
   Stack1={newBarrier(var, exp)| Stmt2|...}
   SymTable1
   Out1
   Heap1
   FileTable1
   BarrierTable1
   ==>
   Stack2={Stmt2|...}
   Out2=Out1
   Heap2=Heap1
   FileTable2=FileTable1
   - evaluate the expression exp using SymTable1 and Heap1 and let be number the result of this evaluation
   BarrierTable2 = BarrierTable1 synchronizedUnion {newfreelocation ->([],number)}
   if var exists in SymTable1 *then*
   SymTable2 = update(SymTable1,var, newfreelocation)
   else SymTable2 = add(SymTable1,var, newfreelocation)
   Note that you must use the lock mechanisms of the host language Java over the BarrierTable in order to add a new barrier to the table.
   c. (1.5p). Define the new statement
   await(var)
   where var represents a variable from SymTable which is mapped to a number

into the BarrierTable. Its execution on the ExeStack is the following:

- pop the statement
- foundIndex=lookup(SymTable,var). If var is not in SymTable print an error message and terminate the execution.
- *if* foundIndex is not an index in the BarrierTable *then*
  print an error message and terminate the execution
  elseif BarrierTable[foundIndex]==(barrier_list,barrier_value) *then*
  if length(barrier_list)==barrier_value *then*
  do nothing
  elseif id of the current thread is in barrier_list *then*
  push back the await statement(that means the current PrgState must wait for the barrier to reach its value)
  else
  - push back the await statement(that means the current PrgState must wait for the barrier to reach its value)
  - synchronized update of BarrierTable for the index foundIndex such that foundIndex is mapped to (barrier_list synchronizedUnion [currentThreadId], barrier_value)

Note that you must use the lock mechanisms of the host language Java over the BarrierTable in order to update an entry of that table.

d. (1p) Extend your GUI to suport step-by-step execution of the new added features. To represent the BarrierTable please use a TableView with three columns location, barrier value and a string representing the list of threads waiting on that barrier.

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 readable log text file.

The following program must be hard coded in your implementation.

new(v1,2);new(v2,3);new(v3,4);newBarrier(cnt,rH(v3));
fork(wh(v1,rh(v1)*10));print(rh(v1);await(cnt)));
fork(await(cnt);wh(v2,rh(v2)*10));print(rh(v2)));
fork(await(cnt);wh(v3,rh(v3)*10));wh(v3,rh(v3)*10));print(rh(v3)));
await(cnt);
print(2000);

The final Out should be {20,2000,30,400}.

2. **(0.5p by default) Implement Sleep statement in Toy Language.**

a. (2.75p). Define the new statement:

sleep(number)

Its execution on the ExeStack is the following:

- pop the statement
- if number== 0 then do nothing
  else push sleep(number-1) 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 following program must be hard coded in your implementation:
v=0;
(while(v<3) (fork(print(v);v=v+1);v=v+1);
sleep(15);
print(v*10)
The final Out should be {0,1,2,30}