CSC 343 Operating Systems, Spring 2021 (Note added Feb. 3 on final page)

Dr. Dale E. Parson, Assignment 1, Implementing and testing a first state machine simulation.
This assignment is due via **make turnitin** from the hideAndSeek2021 directory by **11:59 PM on Friday February 19**. There is a 10% penalty for each day it is late, and I will not accept solutions after I go over my solution in class. I will go over this handout on February 2, and we will have a work session with questions and answers in class on February 4. The February class Zoom recording augments this handout.

The goal of this assignment is to learn how to write an introductory state machine in this semester's STM language. We will simulate a game of hide & seek. There is a README.txt file with questions for you to answer after you have completed the code. **Answers in README.txt are worth 30% of this assignment, so remember that working code is not the end of the requirements**.

Perform the following steps to get my handout. You will code and test on **mcgonagall**, to which you can **ssh mcgonagall** from acad. **Do not change the name of the project directory, since my automated tests depend on that name**.

```
cd  $HOME                # or start out in your login directory
mkdir  OpSys  # All of this semester's work goes under here, skip if you did it before.
cd  ./OpSys
cp  ~parson/OpSys/hideAndSeek2021.problem.zip  hideAndSeek2021.problem.zip
unzip hideAndSeek2021.problem.zip
cd ./hideAndSeek2021
make clean test
```

Testing fails within the handout directory. Some subsequent failures may hang the compiler or simulation, requiring a control-C to abort. Successful **make clean test** finishes in about a second.

```
$ make clean test
make clean test
/bin/rm -f *.o *.class .jar core *.exe *.obj *.pyc __pycache__/*.pyc
/bin/bash -c 'chmod 666 ~parson/tmp/parson_STM*'
chmod: cannot access '/home/kutztown.edu/parson/tmp/parson_STM*': No such file or directory
make: [clean] Error 1 (ignored)
/bin/bash -c '/bin/rm -f *.out *.dif *.pyc junk parsetab.py *.vmlf hideAndSeek2021_crunch.png'
/bin/bash -c '/bin/rm -f *.dot *.png *.jpg testmachine.ck junk.* *.tmp *.log hideAndSeek2021.py'
/bin/bash -c '/bin/rm -f *.crunch ~parson/tmp/parson_STM_*.log parson_STM_*.log Unsafe*.log'
/bin/bash -c '/bin/rm -f rr*.py sjf*.py fcfs*.py plotcrunch.csv *.crunch *_crunch.py *.crunch *_crunch.csv
./__pycache__/*.pyc'
COMPILING hideAndSeek2021
/bin/bash -c "PYTHONPATH=/home/kutztown.edu/parson/OpSys:.:.. /usr/local/bin/python3.7
/home/kutztown.edu/parson/OpSys/state2codeV17/State2CodeParser.py hideAndSeek2021.stm
hideAndSeek2021.dot hideAndSeek2021.py CSC343Compile CSC343Compile"
ERROR, Invalid transition from state loopAndSpawn -> waitForOthersToStart,
waitForOthersToStart not in machine thread.
ERROR, Invalid transition from state initThread -> waitForOthersToStart, waitForOthersToStart
not in machine thread.
ERROR, Invalid transition from state waitForOthersToStart -> waitForOthersToStart,
waitForOthersToStart not in machine thread.
ERROR, Invalid transition from state waitForOthersToStart -> waitForOthersToStart,
```

waitForOthersToStart not in machine thread.
Traceback (most recent call last):
  File "/home/kutztown.edu/parson/OpSys/state2codeV17/State2CodeParser.py", line 807, in <module>
    ptree, symtab = compile(source,dagfilename=sys.argv[2],debugflag=None)
  File "/home/kutztown.edu/parson/OpSys/state2codeV17/State2CodeParser.py", line 740, in compile
    parsetree = yacc.parse(source,debug=debugflag)
  File "/home/kutztown.edu/parson/OpSys/ply/yacc.py", line 331, in parse
    return self.parseopt_notrack(input, lexer, debug, tracking, tokenfunc)
  File "/home/kutztown.edu/parson/OpSys/ply/yacc.py", line 1118, in parseopt_notrack
    p.callable(pslice)
  File "/home/kutztown.edu/parson/OpSys/state2codeV17/State2CodeParser.py", line 565, in
p_statemachine
    states[fromst].addTransition(xition)
KeyError: 'waitForOthersToStart'
make: *** [build] Error 1
[:-) ~/OpSys/hideAndSeek2021]

All of the detailed instructions and grading weights for your code additions appear in **STUDENT** comments in source file **hideAndSeek2021.stm**, which is one of the two files you will change. The other file is README.txt. We will go over your project requirements in class on 2/2.

If you get an error message at run-time that gives an index into __codeTable__ like this:

Traceback (most recent call last):
  File "hideAndSeek2021.py", line 755, in <module>
    main()
  File "hideAndSeek2021.py", line 693, in main
    scheduler.__run__()
  File "/home/kutztown.edu/parson/OpSys/state2codeV17/CSC343Sim.py", line 145, in __run__
    waitingObject.__generator__.__next__() # run() the model
  File "hideAndSeek2021.py", line 500, in run
    if eval(__codeTable__[46],globals,locals):
  File "nofile", line 1, in <module>
IndexError: list index out of range

Just run this decode.py command with that index to see the original source code.

$ **./decode.py hideAndSeek2021.py 46**

__codeTable__[46] = compile('pcb.whoIsHidingHere[pcb.hidingPlaces[**tryHere**]] == None','nofile','eval'),

A successful test run appears as follows.

$ **make clean test**
/bin/rm -f *.o *.class .jar core *.exe *.obj *.pyc __pycache__/*.pyc
/bin/bash -c 'chmod 666 ~parson/tmp/parson_STM*'
/bin/bash -c '/bin/rm -f *.out *.dif *.pyc junk parsetab.py *.vmlf hideAndSeek2021_crunch.png'
/bin/bash -c '/bin/rm -f *.dot *.png *.jpg testmachine.ck junk.* *.tmp *.log hideAndSeek2021.py'
/bin/bash -c '/bin/rm -f *.crunch ~parson/tmp/parson_STM_*.log parson_STM_*.log Unsafe*.log'
/bin/bash -c '/bin/rm -f rr*.py sjf*.py fcfs*.py plotcrunch.csv *.crunch *_crunch.py *.crunch *_crunch.csv
./__pycache__/*.pyc'

```
COMPILING hideAndSeek2021
/bin/bash -c "PYTHONPATH=/home/kutztown.edu/parson/OpSys:.:.. /usr/local/bin/python3.7
/home/kutztown.edu/parson/OpSys/state2codeV17/State2CodeParser.py hideAndSeek2021.stm
hideAndSeek2021.dot hideAndSeek2021.py CSC343Compile CSC343Compile"
/bin/rm -f *.jpg *.png
COMPILING COMPLETED
SIMULATING (TESTING) hideAndSeek2021
/bin/rm -f ~parson/tmp/parson_STM_*.log parson_STM_*.log hideAndSeek2021.log
/bin/bash -c "PYTHONPATH=/home/kutztown.edu/parson/OpSys:.:.. STMLOGDIR=~parson/tmp time
/usr/local/bin/python3.7 hideAndSeek2021.py 2 4 10000 12345 3"
MSG cmd line: ['hideAndSeek2021.py', '2', '4', '10000', '12345', '3'], usage USAGE: python THISFILE.py
NUMCONTEXTS NUMFASTIO SIMTIME SEED|None LOGLEVEL

Scheduler exiting at time 1543 within time limit 10000, simulation has finished.
0.14user 0.03system 0:00.23elapsed 74%CPU (0avgtext+0avgdata 10268maxresident)k
0inputs+584outputs (0major+5224minor)pagefaults 0swaps
/bin/bash -c 'chmod 666 ~parson/tmp/parson_STM*'
/bin/bash -c "PYTHONPATH=/home/kutztown.edu/parson/OpSys:.:.. /usr/local/bin/python3.7
crunchlog.py hideAndSeek2021.log"

DIFFing hideAndSeek2021_crunch.py hideAndSeek2021_crunch.ref
OK: MIN_IamHiding at 20.0% tolerance.
OK: MEAN_IamHiding at 20.0% tolerance.
OK: MAX_IamHiding at 20.0% tolerance.
OK: MIN_IamSeeking at 20.0% tolerance.
OK: MEAN_IamSeeking at 20.0% tolerance.
OK: MAX_IamSeeking at 20.0% tolerance.

# STUDENT, COMMENT OUT NEXT LINE TO SEE THE LOG FILE.
# bash -c '/bin/rm -f ~parson/tmp/parson_STM_*.log parson_STM_*.log hideAndSeek2021*.log'
grep "TAGGED" hideAndSeek2021.log | sort > hideAndSeek2021.out
diff hideAndSeek2021.out hideAndSeek2021.ref
COMPLETED (OK) SIMULATING (TESTING) hideAndSeek2021
```

You can see the successful extracted messages from your program by doing this.

$ **cat hideAndSeek2021.out**

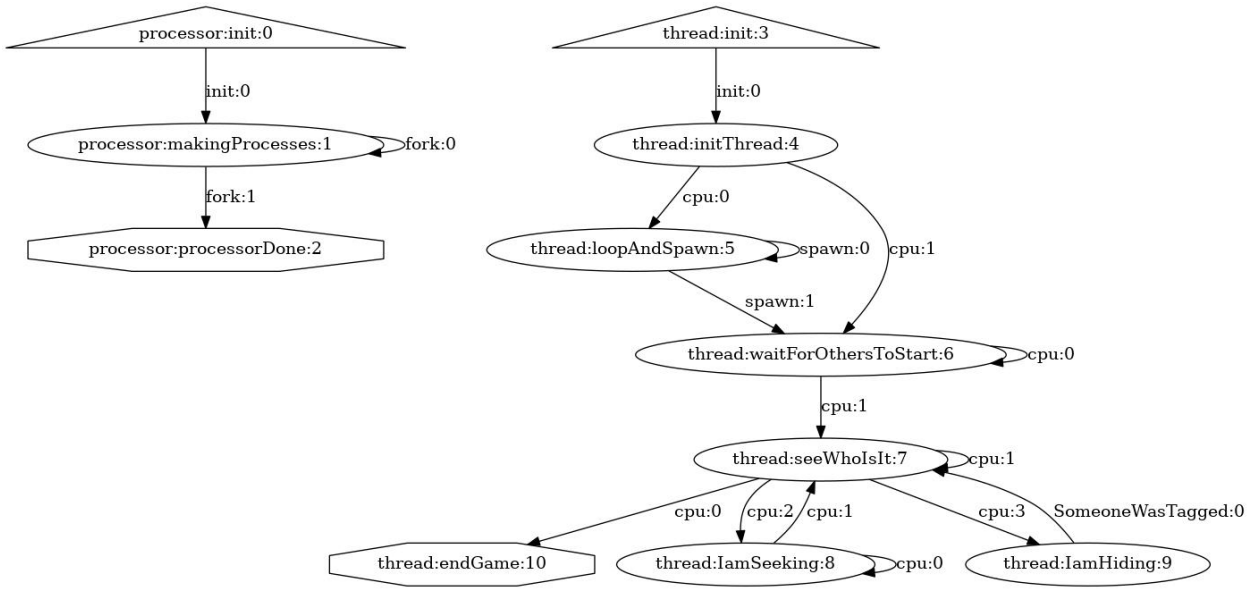
```
000000001543,MSG,thread 0 process 0,0 Thread TAGGED 23 out of 100
000000001543,MSG,thread 1 process 0,1 Thread TAGGED 18 out of 100
000000001543,MSG,thread 2 process 0,2 Thread TAGGED 23 out of 100
000000001543,MSG,thread 3 process 0,3 Thread TAGGED 13 out of 100
000000001543,MSG,thread 4 process 0,4 Thread TAGGED 23 out of 100
```

Any time a COMPILE succeeds, you can look at the graph for your state machine by running **make graphs** and then inspecting https://kuvapcsitrd01.kutztown.edu/~STUDENT/hideAndSeek2021.jpg, where STUDENT is your login ID. If you can't get at it with a browser this way, use WinSCP or FileZilla to copy the JPEG file from your project directory to your local machine. Below is the final, correct graph.

Once **make clean test** passes, **ANSWER THE QUESTIONS IN FILE README.txt** included in this project directory. Follow all instructions in README.txt.

Finally, turn it in by entering **make turnitin** and following the prompt. We do not use the turnin script in this course; instead **make turnitin** turns in the project; it prompts you for a carriage return (Enter) to complete its work.

I will distribute grades via email before the next class after the due date.



NOTE ADDED FEB. 3, emailed to students:

In your assignment, THESE EXPRESSIONS DO NOT WORK:

~~areAllPlayersRunning(pcb.players) == False~~

~~areAllPlayersRunning(pcb.players) == True~~

When at least one position in list pcb.players is None (its initial value), areAllPlayersRunning(pcb.players) returns None.

THIS WORKS, interpreting the returned None as though it were False:

not areAllPlayersRunning(pcb.players)

When all positions in list pcb.players contain pointers to thread objects (pcb.players is fully populated by "pcb.players[tid] = thread ;" in the transitions), areAllPlayersRunning(pcb.players) returns the pointer to the tread contained in the final element of pcb.players.

THIS WORKS, interpreting the returned non-None thread pointer as though it were True:

areAllPlayersRunning(pcb.players)

Those tests are enough. Avoid the "== False" and "== True" comparisons, since Python treats any non-None, non-zero, or non-empty container logically as False, and any None, zero, or empty container as logically True.

Thanks to the student who caught this potential pitfall, and to the students who let me know about permission problems with the STM compiler's **ply** library. Everything is working OK now -- I have seen a working student solution.