

Program 4 Test
CSI4337

Compile and test on wind.ecs.baylor.edu

Compile with: `g++ -g -Wall -ansi -pedantic -lpthread`

-2 points per warning up to -10 for basic warnings

Run with: `valgrind --tool=memcheck --show-reachable=yes --leak-check=full --track-fds=yes ./<executable>`

-2 points for leaked file descriptors up to -10 (ignore fd=0, 1, and 2)

-2 points for each memory leak up to -14. VERIFY submission is responsible for leak

Part I (30 points)

Visually verify proper synchronization for `myRand()`. Program should not limit safe opportunities for parallelism.

Here, they need to protect the printing code with mutual exclusion to make sure two threads won't try to print at the same time. I think the most difficult part will be handling termination. There are a few ways to deal with this properly, but there should be 1) no busy waiting, 2) exactly the right number of topics printed and 3) no a priori division of labor.

Pretest

- Much of the code is mine. Don't consider my code
- Kill any student yields and sleeps
- Add `usleep(5)` and print of thread ID before ad lib print. Verify multithreaded
- Verify all system calls are properly tested and reported: -2 per call up to -5
- Verify good division of prints. You don't know how long each print will take a priori; however, all n threads should complete as simultaneously as possible: -10 points
- Verify efficiency. You should not call `noun()`, etc. inside critical section since those functions may take a long time. Only use mutual exclusion for printing

Part II (70 points)

- Most of the code is mine. Only `GameMonitor` code should be new
- Uses only integers? -3 points
- Only use `notify()`, not `notifyAll()`? -2 points
- Handles spurious wakeups? -10 points
- Add `test()` call to end of `decide()` method
- Run for 200,000 games; verify no deadlock or test failure
- Add empty `notify()` implementation; run; and verify program deadlocks after first play of second game

For both parts

- Take off points for poor coding practices
- Verify correct use of system calls, including test of return values. ALL FUNCTION RETURNS SHOULD BE CHECKED!
- Verify use of system error messages (e.g., `errno`, `perror`, `strerror`, exception messages) when available. Identify overuse

-2 unused include/import

-10 for poor commenting

-2 for each unjustified numeric constant

-5 bad submission

-1 for each section of commented code

-5 if violate coding convention