Steven Tardo
25 February 2015
CSCI 4401
Homework 1

1) At time 22 P1 is blocked for I/O, P3 is blocked for I/O, P5 is ready or running, P7 is blocked for I/O, and P8 is ready or running. At time 37 P1 is ready or running, P3 is ready or running, P5 is swapped out, P7 is blocked for I/O, and P8 is ready or running. At time 47, P1 is ready or running, P3 is ready or running, P5 is ready or running, P7 is blocked for I/O, and P8 has terminated.

2) An intermediate policy that could balance priority and performance could be to execute processes in the Ready state until the process with the highest priority in the Ready/Suspend state has been swapped, execute that process, and continue with this policy as needed.

3) Depending on whether the forked process completes first or second, the output from the forked process would be printed followed by the child pid, or vice versa.

4) This allows multithreaded programs to run faster because other threads in the program can continue processing while the blocked process is blocked, where as in a single-threaded program the entire program is blocked until the single thread is ready again.

5) a) This program counts the number of positive elements in a list.
   b) There shouldn't be any problems because thread A is not performing any writing operations on the list, only reading. Only thread B is writing.

6) A potential problem could occur is thread A increments r before thread B has a chance to set global_positives to r, or if thread A sets r to global_positives before thread B has a chane to set global_positives to r.

7) a) This program creates a thread that prints a period, then the main program prints an "o", and then the next thread executes.
   b) The output is not correct because there should be one period followed by one "o" every time, and myglobal should equal 40 since each thread increments it once and the main program increments is twenty times.

8) a) If P1 performs the decrement and writes the value to memory, and P2 performs the decrement and stores the value to memory, the value of x will be 8. Then P1 performs the increment and stores the value to memory and the value of x is 9. It then checks the value of x and determines that it is not equal to 10. However, before it gets a chance to print, P2 takes over, increments the value to 10, then loses control. P1 takes back over and prints the statement x is 10.
   b) P1 loads 10 from memory, decrements it, and stores it back in memory. P2 then loads 9 from memory, decrements it to 8, but doesn't get a chance to store it in

memory. P1 then loads 8 from memory, increments it, and stores 9 in memory. P2 takes over and overwrites 9 in memory with 8. P1 then loads 8 from memory, increments it, but doesn't get a chance to store it. P2 loads 8 from memory, increments it 9, and stores it in memory. P1 then continues, storing 9 to memory, and both processes perform their print statements, which will be x is 9. This set of sequences happens again, except the starting value is now 9, so once all the steps are complete x is 8 will be printed.

9) a) If the program executes perfectly, 100 is the maximum value that tally can hold. However, suppose Process 1 loads 0 into its register, increments the value, but doesn't get a chance to store it. Process 2 then takes over, loads tally, which is still at 0 since Process 1 still hasn't had a chance to store it yet. Process 2 performs 49 increments and stores the value in memory. Process 1 then takes over, and overwrites 49 with 1. Process 2 takes over and performs its last increment, bringing tally 2, but doesn't get a chance to store it. Process 1 takes over, finishing its task and writing 50 to memory. However, Process 2 still has to store its value in memory, which is 2. Once this is complete both processes are finished and tally has a value of 2.

   b) If no other processes touch tally and the program executes as in the last part of the problem, the minimum value could still possibly be 2. However, for example if there are ten processes, and all ten processes execute completely in order one after the other, then tally will be incremented 50 times by each process, bringing the total to 500.

10) Both definitions will have the same effect on a program. Although a semaphore can never take on a negative value in the definition from the problem, there is an explicit if statement that says to place a process on the ready list regardless of the value of s.count as long as there is at least one process blocked. Figure 5.3's definition says the same thing, but only blocks processes if s.count is less than 0, while the problem definition only decrements s.count if there are no processes blocked.