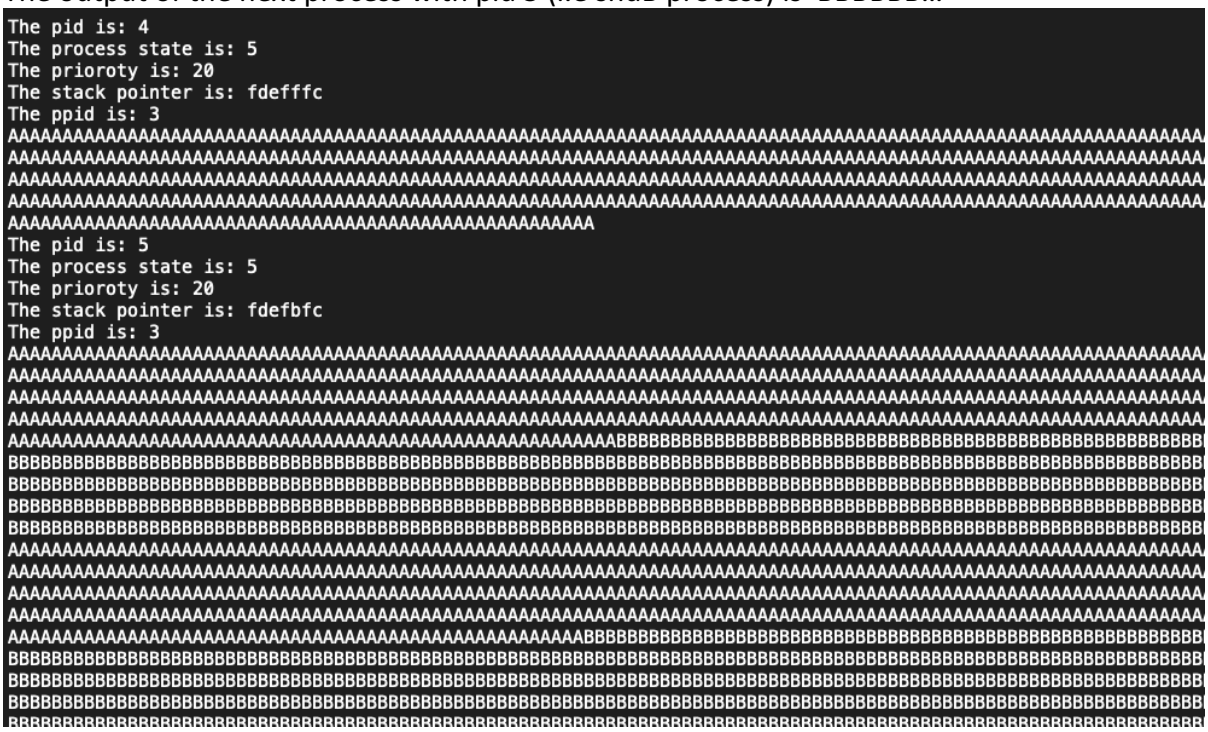
CS 354 Lab 2 Answers

3.2

a. An alternate output of A's and B's is observed.
The output of the first process with pid 4 (i.e sndA process) is AAAAAA…
The output of the next process with pid 5 (i.e sndB process) is BBBBBB…



Explanation: After the sndA process is called the output of multiple As is observed. The next process is created, but now both the processes have the same priority hence using round robin manner a time of 50ms is allocated to sndA thus outputting multiple As again on the screen. EOI command is issued, clkhandler() is called and when the 50ms ends the process is rescheduled and another 50ms is allocated to sndB and an output of multiple Bs is observed and this alternate process continues forever because of an infinite loop.

b. For 25 ms the output is similar to the output produced for 50ms QUANTUM. The only difference being that **less** number of As and Bs are printed before the process is rescheduled after the 25ms time budget ends as compared to 50ms QUANTUM.
For 75 ms the output is similar to the output produced for 50ms QUANTUM. The only difference being that **more** number of As and Bs are printed before the process is rescheduled after the 75ms time budget ends as compared to 50ms QUANTUM.

c. The output of multiple As is produced from the sndA process with pid 4 and priority 30. The next process (sndB) with is never made because of the infinite loop in sndA function and because sndB has a lower priority hence the scheduler never gets to that process.

d. First the output of multiple As is produced from the sndA process with pid 4 and a priority of 20. Then the next process with pid 5 and priority 30 is produced and it outputs infinite Bs because of the infinite loop. It doesn't go back to the process with pid 4 because it has less priority than process with pid 5 and process with pid 5 will never end because of infinite loop hence the output.

e. Observation: Output of multiple As is seen and then when the process with pid 5 is created, an output of multiple As is observed till its time slice ends, this is followed by multiple Bs. Then welcome() is called from initialize.c and the welcome message is printed.
Explanation: This happens because the processes with pid 4 and pid 5 finish after their allocated tie runs out and the null process takes over since it has a higher priority (0 > -3). The null process continues to run. Basically, processes with pid 5 and pid 6 get terminated after running once during their time allotted.

BONUS:
I have implemented a system call called procinfo which takes an argument of pid of type pid32 and displays the information of the process state, priority, stack address, parent pid and time slice. I implemented this system call as I thought it would be useful in debugging and finding more information about processes.