Name: ___Justin Weigle_________

1. Multiprogrammed computers depend on the cpu burst – io burst cycle. Explain why. Write a C function, no matter how trivial, that does not behave this way:

Multiprogrammed computers execute a process until it must wait, usually for the execution of an I/O request. Therefore if a process never hits an I/O request, it doesn't allow other programs to run.

```
void do_a_loop () {
   int i = 0;
   while (1) {
      i += 1;
   }
}
```

2. The job of the CPU scheduler is to pick the next process to run. Truly, it is not possible to make a perfect scheduler. Explain why.

There are way too many criteria involved in picking a scheduling algorithm that fits every situation. The primary criteria are: CPU utilization, throughput, turnaround time, waiting time, and response time. Each algorithm can favor one class of processes over another based on the listed criteria.

If you get a lot of processes that have very short CPU times and a couple processes with long CPU times, the long running processes may wait a very long time before getting to run under Shortest-Job-First Scheduling.

On the other hand, under Priority Scheduling, processes with low priority might wait a very long, and possibly even indefinite, amount of time before getting to run if there are many high priority tasks.

3. Preemptive scheduler depend on ___Interrupts___ to get control back from the currently running user program. Why can't the OS just take control back on its own?

Preemptive scheduling can result in race conditions. Since this can cause severe problems in regards to data being shared across processes, there are mechanisms such as mutex locks that prevent said race conditions by temporarily disabling interrupts.

4. The text described two types of context switches – voluntary and involuntary. What triggers these different types?

   Voluntary - process gives up control of the CPU because it requires a resource that is currently unavailable.
   Nonvoluntary - CPU is taken away from a process, time slice expired/preempted by a higher priority process.

5. Consider the following processes and then answer the following questions using *First-Come, First Served* scheduling:

Process	*Arrival Time*	*Burst Time*
P1	*0*	*1.5*
P2	*1*	*0.5*
P3	*3*	*2.5*
P4	*4*	*1.5*

a. Show the Gantt chart for these four processes (see text for example):

P1	P2		P3	P4

0 1.5 2 3 5.5 7

b. Compute the average *CPU utilization:*

$1 - p^n$ Utilization = 99.96%
p = 1/7
n = 4

c. Compute the average *turn-around time:*

P1 → 1.5
P2 → 0.5 + 0.5
P3 → 2.5
P4 → 1.5 + 1.5

1.5 + 1 + 2.5 + 3 = 8
8 / 4 = 2
Average turnaround = 2

d. Compute the average *wait-time:*

P1 → 0
P2 → 0.5
P3 → 0
P4 → 1.5

0 + 0.5 + 0 + 1.5 = 2
2 / 4 = 0.5
Average wait time = 0.5

6. Consider the following processes and then answer the following questions using *Shortest-Job First* scheduling:

Process	*Burst Time*
P1	*1.5*
P2	*0.5*
P3	*3.5*
P4	*4.5*

a. Show the Gantt chart for these four processes (see text for example):

P2	P1	P3	P4

0 0.5 2 5.5 10

b. Compute the average *CPU utilization:*

$1 - p^n$

$p = 0$

$n = 4$

Utilization = 100%

c. Compute the average *turn-around time:*

P1 → 0.5 + 1.5
P2 → 0.5
P3 → 2 + 3.5
P4 → 5.5 + 4.5

2 + 0.5 + 5.5 + 10
10 / 4 = 2.5
Average turnaround = 2.5

d. Compute the average *wait-time:*

P1 → 0.5
P2 → 0
P3 → 2
P4 → 5.5

0.5 + 0 + 2 + 5.5 = 8
8 / 4 = 2
Average wait time = 2

7. Considering you answer in question 2, suppose we have a job that has the following *actual* bursts. Using the *exponential average*, find an *alpha* value that minimizes the RMSQE between *predicted* and the *actual* burst:

   T_1 = 0.5, 1.5, 2.0, 1.5, 4, 2.5, 4, 2.5, 3.5

a. Show your results for computing alpha:

   RMSQE is 1.153 when alpha is 0.622 and initial T_n is set to 0

   Predictions are as follows:
   0
   0.5
   1.499
   1.999
   1.5
   3.998
   2.501
   3.999
   2.501

b. Next, using your alpha value from part a, make predictions and compute the RMSQE for a job with following burst-times. Clearly state whether your accuracy was better or worse.

   T_2 = 1.5, 2.5, 3.5, 4.5, 5.5, 3.5, 1.5, 2.5, 3.5

   RMSQE is 1.554 which is WORSE accuracy by a bit

   Predictions are as follows:
   0
   0.933
   1.908
   2.898
   3.894
   4.893
   4.027
   2.455
   2.483

8. Why do most OSes today use multi-level scheduling?

Multilevel scheduling is the most general CPU scheduling algorithm. It is able to be adapted to many different systems and situations. Since today there are so many different machines and situations that an operating system can be running in, we need the most general scheduling algorithm in order to fit everyone's needs.