1. Basic Concepts I (15 pts) Give brief answers.

a. In a multiprogrammed uniprocessor system, how does control pass from an executing user process to the operating system?

b. How does control pass from the operating system to a user process?

c. The Process Control Block (PCB) is a key operating system data structure. Why is the PCB so important?

d. The Program Counter (PC) is a key CPU register. What happens to the PC when an I/O interrupt occurs?

e. List one advantage and one disadvantage of the Shortest Job First scheduling algorithm.
f. List one advantage of semaphores and one disadvantage of semaphores.

g. Define operating system!!! (Write a succinct high quality definition. Think before you write. Do not core dump a bunch of facts.)

2. More Basic Concepts II: Frequency of Events (20 pts)

For the following pairs of events or situations, quantify the frequency (values) of the two events. Use the abbreviations shown in the table. **Explain each answer briefly but clearly.**

<table>
<thead>
<tr>
<th>&lt;</th>
<th>&lt;=</th>
<th>&gt;</th>
<th>&gt;=</th>
<th>=</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strictly less than</td>
<td>Less than or equal</td>
<td>Strictly greater than</td>
<td>Greater than or equal</td>
<td>Equal or approx. equal</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

a. # of processes in ready queue ___________ # of processes in blocked queue

b. Length of Round Robin (per process) time quantum ___________ next predicted CPU burst using exponential averaging

c. # times Process X is in ready state ___________ # times Process X is in blocked state
d. # times the scheduler is called ___________ # times an interrupt occurred

e. # times a clock interrupt occurred ___________ # times an I/O interrupt occurred

f. # times P(mutex) is executed ___________ # times V(mutex) is executed to protect critical section code

g. # of processes in scheduler’s blocked queue ___________ # of processes in semaphore S1’s blocked queue

h. In the Dining Philosophers Problem,
# philosophers ___________ # chopsticks

i. Over a 24 hour period, total time CPU is executing user code ___________
total time CPU is executing OS code.

j. # times you have watched the Avatar ___________ # of times Ginnie Lo has watched Avatar ___________# times Edsgar Dijkstra has watched Avatar.
3. **Synchronizing Siblings in a Tree of Processes (15 pts)**

Write pseudocode using fork and exec in which the parent forks off four children, and each of these children fork off four children to yield a tree of processes. All processes are trying to enter critical sections. Your code must ensure that the (four) direct descendants of a given parent node cannot enter critical section concurrently. Other combinations of processes are allowed to freely execute without synchronization.

How many semaphores do you need? _____________
4. Exponential Averaging (15 pts)

(a) What is the purpose of exponential averaging and why does it work pretty well?

(b) Suppose alpha = 1/4. In the estimate of TAU(n+1) what is the ratio of the weights given to two actual bursts that are 4 iterations apart, such as t(8) and t(4), or t(7) and t(3). Show your work for partial credit.

\[
\frac{\text{weight on } t(k)}{\text{weight on } t(k-4)} = ???
\]

(c) What metric would you propose for evaluating the accuracy of exponential averaging predictions? You can use the one from your group solution in HW2 or make up a new one. Give a mathematical definition here.
5. Semaphores (15 pts)

a. Below is the code for a blocking implementation of P() and V() with semaphores.

```c
typedef struct { int value; /* counter */
                 struct process list; /* list of processes */ }
               semaphore;

P(semaphore *S) {
    S->value--;
    if (S->value < 0) {
        Add this process to S->list;
        block();
    }
}

V(semaphore *S) {
    S->value++;
    if (S->value <=0) {
        remove a process P from S->list;
        unblock(P);
    }
}
```

For P() and V() to function correctly, they must execute atomically. This means that the P() and V() code are themselves critical sections relative to each other.

(i) Why must they execute atomically?

(ii) In the space below, write code to ensure that P() and V() execute atomically. You can write “code for P” and “code for V” as shorthand (need not copy all the code above).

b. If semaphore S is initialized to 5, and 17 processes have executed P(S) but no processes have executed V(S), what is the value of S->value? ___________ and how many processes are in the list S->list? ______________

c. When a process in the semaphore blocked queue is signaled that it can now enter critical section, to where does the OS next move the PCB? Circle all that apply.

Scheduler blocked queue  Scheduler ready queue  Running Process
Semaphore ready queue  Semaphore busywait queue  Semaphore hotel
6. Scheduler Pseudo-code (20 pts)

Write pseudo-code for a Scheduler that computes normalized turnaround time for each process. Recall that normalized turnaround time is defined as

\[ \text{NTT} = \frac{\text{turnaround time}}{\text{runtime}} \]

Your code should accomplish the following:

- In each case append the running process to the appropriate queue.
- Compute process runtime = total time that the process is executing as the running process.
- Compute process turnaround time = total time from when the process is created to when it exits and becomes a zombie.
- When the process exits (becomes a zombie) compute and print out the process ID and the normalized turnaround time for that process.

The basic code has been provided for selecting next process (round robin) and context switch. You are to add your new code in the spaces provided. You need not use all the lines provided (the purpose of the lines is to keep your answers neat so I can grade them more easily). Add comments next to your code to help explain your solution.

You are free to define any new fields in the PCB, any new global variables or any new procedures. Define these here, give initial values, give short explanation:

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------
Scheduler () {

If (called by fork()) { /* This call was made by the parent process, the current running process. It passes a pointer to CHILD_PCB */

Running_Process = Head(Ready_Queue); /* get next process */
start_time = read_time(); /* record its start time */
if (Running_Process != NIL) Context_Switch (Running_Process, DELTA);
else IDLE(); /* start next process or idle until next interrupt */

else if (called by I/O routine) { /* running process requested I/O */

Running_Process = Head(Ready_Queue); /* get next process */
start_time = read_time(); /* record its start time */
if (Running_Process != NIL) Context_Switch (Running_Process, DELTA);
else IDLE(); /* start next process or idle until next interrupt */
} else if (called by I/O interrupt handler) {

Running_Process = Head(Ready_Queue); /* get next process
start_time = read_time(); /* record its start time
if (Running_Process != NIL) Context_Switch (Running_Process, DELTA);
else IDLE(); */ start next process or idle until next interrupt

} else if called by timer interrupt handler {

Running_Process = Head(Ready_Queue); /* get next process
start_time = read_time(); /* record its start time
if (Running_Process != NIL) Context_Switch (Running_Process, DELTA);
else IDLE(); */ start next process or idle until next interrupt
} else if called by exit() { /* zombie */

Running_Process = Head(Ready_Queue); /* get next process
start_time = read_time(); /* record its start time
if (Running_Process != NIL) Context_Switch (Running_Process, DELTA);
else IDLE(); /* start next process or idle until next interrupt

} /* end scheduler pseudo code */

7. Feedback

a. This exam was: _____EASY _____SO-SO _____HARD

b. How much did you study for this exam?

_____ < 2 hours _____ 2-4 hours _____ 4-8 hours _____ > 8 hours

c. Would you prefer to have two shorter midterms in weeks 4 and 7, rather than one midterm in Week 5 or 6. _____ YES _____ NO _____ DON’T CARE