1. **Short Answer**

a. The Process Control Block (PCB) is the most important data structure managed by the OS. From the following list, circle those items that very likely would be stored in the PCB.

<table>
<thead>
<tr>
<th>PID (process ID)</th>
<th>PC (program counter)</th>
<th>CPU registers</th>
<th>process code</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU status info</td>
<td>link to next PCB</td>
<td>process state (ready, blocked, running)</td>
<td></td>
</tr>
<tr>
<td>test-and-set variable</td>
<td>semaphore used by this process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>previous predicted CPU burst</td>
<td>password for the user that created this process</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. The text and lectures discuss three standard ways of evaluating the performance of operating systems (scheduling algorithms in particular). The three methods are queueing theory analysis, simulation, and empirical experiments. For each of the dimensions of performance evaluation listed below, circle the method which is **best** for that dimension.

   (i) **accuracy** - best in the ability to accurately predict the future performance of the system when deployed

      queueing theory  simulation  empirical experiments

   (ii) **cost/implementation overhead** - best (least) amount of design, programming, equipment, and effort to do the performance analysis

      queueing theory  simulation  empirical experiments

   (iii) **flexibility** - maximum ability to model a wide variety of algorithms and workloads (not taking cost into account)

      queueing theory  simulation  empirical experiments
c. Leslie Lamport developed the Bakery Algorithm. What problem does it solve?

d. Name two advantages of semaphores.

e. How is atomicity of the wait(0 and signal() kernel code guaranteed?

f. How is the atomicity of the test-and-set() hardware instruction guaranteed?

g. What is the difference between the Unix fork and exec kernel calls?

h. Write the formula for normalized turnaround time. What is the theoretical minimum value of normalized turnaround time?
2. Process State Transitions
In all these problems assume a multiprogrammed uniprocessor system with several processes running along with the one that the problem focuses on.

a. Indicate what state(s) a process occupies if it is starved by the SJF algorithm by drawing arrows to show the state transitions. If only one state, just circle that state.

ready  running

blocked

b. Indicate what state(s) a process occupies if it is livelocked by a faulty busywait solution to the critical section problem, by drawing arrows to show the state transitions. If only one state, just circle that state.

ready  running

blocked

c. Indicate what state(s) a process occupies if it is deadlocked by a faulty solution to the critical section problem that uses blocking semaphores. Draw arrows to show the transitions or if only one state, just circle that state.

ready  running

blocked

d. Indicate what state(s) a process occupies immediately before and after it executes wait(sema) when the semaphore value is 0. Draw states arrows to show the state transitions or if only one state, just circle that state.

ready  running

blocked
e. Indicate what state(s) a process could occupy while in critical section by listing those states.

- ready
- running
- blocked

3. Exponential Averaging

The Sun Computer Company OS uses exponential averaging with alpha = 0.1 to predict the next CPU burst for its SJF scheduling algorithm. The Moon Computer OS uses a technique in which the next predicted CPU burst is simply the average of all previous actual bursts. More precisely,

Sun Company’s formula: \( \text{Tau}(n+1) = \alpha \times t(n) + (1-\alpha) \text{Tau}(n) \quad \alpha \text{ in } [0,1] \)
Moon Company’s formula: \( \text{Tau}(n+1) = \frac{1}{n} \times \sum_{i=1}^{n} t(i) \)

a. In the computation of the fifth predicted burst \( \text{Tau}(5) \), how much weight is given to \( t(2) \) under each system? Circle and label final answers. Show work for partial credit.

b. How many bursts must be completed before Sun’s exponential average system gives more weight to \( t(n) \) in \( \text{Tau}(n+1) \) than Moon’s straight average system? In other words, what is the smallest value of \( n \) for which this occurs? Your answer can be a formula. Circle and label final answers. Show work for partial credit.
Using Semaphores for Registration in CIS 415 (10 pts)

Define appropriate semaphores and write code using wait() and signal to coordinate students’ code so that at most 30 students take CIS 415 simultaneously. In addition, you code must ensure that the student has completed CIS 314 and CIS 313 before registering for CIS 415.

The structure for your code should be:

```c
take_CIS415() {
    check 314 pre-req
    check 313 pre-req
    take CIS 415
}
```

6. Synchronization and Scheduling (20 pts)

Write pseudo-code for a Scheduler that gathers statistics on how long a process is blocked waiting on semaphores. For simplicity, assume a given process only uses one semaphore during it’s lifetime. Use the data structures and pre-defined procedures given to you in Homework 1 (reproduced for you on the yellow sheet).

Your code should accomplish the following:

- Whenever a process blocks on a semaphore, compute how long it stays blocked on that semaphore, and write the information to the statistics file (semaphore name and time blocked before signaled)
- When the process exits (becomes a zombie) print out the total time blocked on the semaphore over the lifetime of the process.
- Assume that the scheduler is passed the following information when called from a wait() or signal() kernel call: the name of the semaphore and the process ID of the calling process.

The basic code has been provided for you. 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 () {

/* The cases for I/O routine call, I/O interrupt handler have been omitted */

If (called by wait synchronization kernel call) {
    /* This call was made by process PID on semaphore SEMA to block the process */

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

} else if (called by signal(sema) kernel call) {
    /* This call was made by process PID on semaphore SEMA to unblock a process waiting on SEMA */

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

} else if (called by timer interrupt handler) { /* after time quantum expired

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

} else if (called by finished process) { /* process is done

/* write the total synch block time to the statistics file */

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

}