1. What is an Operating System? (Points each, 4 problems)

2. What is the difference between blurring and smoothing? (6 points)

3. Which process scheduling algorithm(s) can NOT be both preemptive and non-preemptive? (3 points)

4. Process scheduling algorithms with a few modifications become (or behave) like other scheduling algo-

5. Priority scheduling with highest priority being the job that has been in the system the longest is like-

6. Round Robin - always preemptive (between turns)

7. FCFS - never preemptive

8. SF is a special case of what other algorithm?

9. Priority, where the shortest job has priority.

10. FCFS

11. Process scheduling with highest priority being the job that has been in the system the longest is like-

12. What is the difference between blurring and smoothing? (6 points)
3. RR

\[ P_1 = 3 \text{ minutes} \]

2. Precipitate Priority

\[ \text{Precipitate} = \frac{1}{2} \]

1. Precipitate SJF

\[ \text{SJF} = \frac{1}{2} \]

Where is the execution waiting time for each of the 3 scheduling algorithms in part a)?

---

3. RR (Round-Robin) (Quantum = 1)

2. Precipitate Priority (lower number is higher priority)

1. Precipitate SJF (use time remaining in all scheduling decisions)

---

a) Fill in the 3 charts below (Process # on y-axis, time on x-axis) illustrating the execution of the processes.

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P3</td>
<td>12</td>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>
You need a context switch to start the first job. Taking into account context switch time and assuming that the context switch happens at the end of each job, you need a context switch.

Given the following arrival times, with quantum 2 and so on. A preemptive given the following three processes, with quantum 2, process 1 is done before any in the queue. In other words, all processes are done before any in the queue. Also assume that priority is given to processes in queues with lower priority of 2. Assume that a context switch requires 1 ms, and that it doesn't take any additional time to do so. The quantum in the next is 2, then 4, then 8, 16, 32, 64, and so on. The quantum incrementing as you go.

Assume you are using multi-level feedback queues, where the quantum in the first is 1 ms (millisecond).
Consider the following diagram and section of code which creates a pipeline:

```
1 #define STD_INPUT 0
2 #define STD_OUTPUT 1
3 #define pipeline (char *proc1, char *proc2)
4 { pid = fork();
5   if (pid == 0) {
6     pipe (fd[0]);
7     close (fd[1]);
8     exec (proc1);
9     close (fd[0]);
10    close (STD_INPUT);
11    close (STD_OUTPUT);
12    close (fd[1]);
13    execl (proc2, 0, 0);
14    close (fd[0]);
15    close (fd[1]);
16    close (STD_INPUT);
17    close (STD_OUTPUT);
18    close (std[0]);
19    dp
20 } else {
21   // Child
22   // create a pipe
23   // for pipe file descriptors
24   // Assume proc1 and proc2 are the 2 processes
25   // for std input and std output
26   // write to pipe
27   // read from pipe
28   // return
29   // 16, close
30   // 17, close
31   // 15, close
32   // 14, close
```

Essentially a reverse pipe B is piped into the input of program A,
The result is that processes take turns on a Round Robin. Each dotted line represents the short-term scheduler running.

What is the algorithm that results from being greater than or equal to \( \alpha > 0 \)? (Hint: draw a graph showing two processes arriving at the same time, showing what happens to their priorities over time.)

The short-term scheduler runs regularly to ensure the process with the best priority gets to run. Priority changes at a rate \( \beta \). All processes are given a priority of 0 when they enter the ready queue. CPU (in the ready queue, but not running) its priority changes at a rate of \( \alpha \). When it is running, its priority increases proportionally to the load. Larger priority numbers imply higher (more favorable) priority. When a process is waiting for the CPU, the following preemptive priority scheduling algorithm based on dynamically changing priority...
5. Most of the Shortest Job First (SJF) Scheduling we have seen involves scheduling based on the entire burst time until the job is finished. It is more accurate to consider the various turns a job gets in the CPU before it is completed. Since we don’t know the burst time for a particular turn until after the job runs, we make estimates using Exponential Averaging, where the system assigns a default value for the very first scheduling decision. Assume an initial estimate \( T_0 \) of 10, with the subsequent actual bursts \( t_n \) as shown in the table below. Assume \( \alpha = 1/2 \).

a) Complete the table below by computing subsequent burst time estimates \( T_{n+1} \), writing them in the spaces provided for each process.

```
<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Initial burst Estimate ( T_0 )</th>
<th>( t_0 )</th>
<th>( t_1 )</th>
<th>( t_2 )</th>
<th>Total Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td>17</td>
<td>12</td>
<td>30</td>
</tr>
<tr>
<td>P2</td>
<td>15</td>
<td>6</td>
<td>16</td>
<td>5</td>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>P3</td>
<td>33</td>
<td>6</td>
<td>12</td>
<td>7</td>
<td>7</td>
<td>19</td>
</tr>
</tbody>
</table>
```

b) Complete the chart below, showing which process is running at each point in time, using preemptive SJF Scheduling. If there is a tie, use FCFS where the lower numbered process goes first. Note that the actual burst time may be longer than the estimated burst.

*** Important *** Preemption decisions are made on the most recent estimate for each job, not on how long it actually runs or on the time remaining. If and when a job is preempted, it must finish its turn before a new burst estimate is calculated.

e) What is the mean wait time?

\[
\begin{align*}
\text{Wait time} &= \text{End time} - \text{arrival time} - \text{burst time} \\
\text{P1:} & \quad 70 - 0 - 30 = 40 \\
\text{P2:} & \quad 65 - 15 - 21 = 29 \\
\text{P3:} & \quad 60 - 33 - 19 = 8 \\
\end{align*}
\]

\[
\bar{77}/3 = 25.6
\]