Homework I

1. Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5 and X. In what order should they be run in order to minimize average waiting time? (Hint: Though the answer is simple, you need to consider all cases).

In class we’ve discussed that shortest job first leads to minimum average waiting time. See class text, p 159. You don’t need to prove it for this question, just remember that fact. In that case the answer is

a. $0 < X \beta 3$: X, 3, 5, 6, 9
b. $3 < X \beta 5$: 3, X, 5, 6, 9
c. $5 < X \beta 6$: 3, 5, X, 6, 9
d. $6 < X \beta 9$: 3, 5, 6, X, 9
e. $X > 9$: 3, 5, 6, 9, X

2. This problem deals with real time systems with N CPU’s. Events that real time systems have to respond to are classified as periodic or aperiodic. Consider the first case where jobs occur at regular intervals: Let there be m periodic events and event $i$ occurs with period $P_i$ and $C_i$ seconds of CPU time are required to handle each event, derive an inequality that can be used to show the conditions under which the periodic load can be handled. (Hint: Define CPU utilization, ignore context switch overhead.)

Utilization is defined as $\sum_{i=1}^{m} \frac{C_i}{P_i}$ and this must be less than or equal to N, i.e., CPU utilization cannot exceed the number of CPU’s available, and we have assumed N CPU’s.

3. For the bakery algorithm we define a processor is in the bakery from when it sets choosing[$i$] = false till it fails or leaves the critical section. The two lines before it is in the CS it called as being in the doorway. Prove the following assertions:

   a. Assertion 1: If processors i, k are in the bakery and i entered the bakery before k entered the doorway, then number[i] < number[k].

   b. Assertion 2: If processor i is in its critical section, processor k is in the bakery, and k .not equal. i, then (number[i] , i) < (number[k] , k).

   **Assertion 1**: If processors i, k are in the bakery and i entered the bakery before k entered the doorway, then number[i] < number[k].

   **Proof**: By hypothesis, number[i] had its current value while k was choosing the current value of number[k]. Hence, k must have chosen number[k] >= 1 + number[i].

   **Assertion 2**: If processor i is in its critical section, processor k is in the bakery, and k .not equal. i, then (number[i] , i) < (number[k] , k).

   **Proof**: Let TL2 be the time at which processor i read choosing[k] during its last execution of L2 for j = k, and let TL3 be the time at which i began its last execution of L3
for \( j = k \), so \( TL_2 < TL_3 \). When processor \( k \) was choosing its current value of \( \text{number}[k] \), let \( T_e \) be the time at which it entered the doorway, \( T_w \) the time at which it finished writing the value of \( \text{number}[k] \), and \( T_c \) the time at which it left the doorway. \( T_e < T_w < T_c \). Since \( \text{choosing}[k] \) was equal to zero at time \( TL_2 \), we have either a) \( TL_2 < T_e \) or b) \( T_c < TL_2 \). In case a), \textbf{Assertion 1} implies \( \text{number}[j] < \text{number}[k] \) so \textbf{Assertion 2} holds.

4. A computer has six tape drives, with \( n \) processes competing for them. Each process may need two drives. For which values of \( n \) is this system safe?

With \( n = 6 \), i.e., six processes, each one holding one tape drive, and wanting another one, we have a deadlock. For \( n < 6 \), the system is deadlock-free (check it out for yourself).

5. Consider a system consisting of \( m \) resources of the same type, being shared by \( n \) processes. Resources can be requested and released by processes only one at a time. Show that the system is deadlock free if the following two conditions hold:
   a. The max need of each process is between 1 and \( m \) resources
   b. The sum of all max needs is less than \( m + n \)

This is problem 8.9 in the textbook. You should have a solutions manual available and should refer to it. A) \( \sum_{i=1}^{n} \text{Max}_i < m + n \). B) \( \text{Max}_i \geq 1 \) for all \( i \). (\( \text{Need}_i = \text{Max}_i - \text{Allocation}_i \)). If there exists a deadlock then C) \( \sum_{i=1}^{n} \text{Allocation}_i = m \). Use A) to get

\[
\sum_{i=1}^{n} \text{Allocation}_i + \sum_{i=1}^{n} \text{Need}_i = \sum_{i=1}^{n} \text{Max}_i < m + n .
\]

Use C) to get

\[
m + \sum_{i=1}^{n} \text{Need}_i = \sum_{i=1}^{n} \text{Max}_i < m + n \text{ or } \sum_{i=1}^{n} \text{Need}_i = \sum_{i=1}^{n} \text{Max}_i < n .
\]

This implies that there exists a process \( P_i \) such that \( \sum_{i=1}^{n} \text{Need}_i = 0 \). Because of B) it follows that \( P_i \) has at least one resource that it can release. Hence the system cannot be in a deadlock state.

6. If an instruction takes 1 microsec and a page fault takes an extra \( n \) microsec, derive a formula for the effective instruction time if page faults occur every \( k \) instructions (on average).

Every \( k \) instructions we add \( n \) microseconds therefore for 1 instruction we add \( n/k \) microseconds. The effective instruction time on average is \( 1 + n/k \).

7. The figure below shows a virtual address space from 0 to 64K and 32K of physical memory. There are 16 pages and 8 frames and transfers between memory and disk are in pages. Give the physical address corresponding to the following virtual addresses: a) 20 b) 4100 c) 8300
8212 b) 4100 c) 24684

8. Solve problem 10.10 in our textbook.

a) 5000 page faults. b) 50 page faults.


<table>
<thead>
<tr>
<th># of frames</th>
<th>LRU</th>
<th>FIFO</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>16</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>