**Q1** The Peterson’s solution algorithm for mutual exclusion is under consideration. Trace the algorithm for two processes if clock interrupt occurs at position 3.

```c
int turn;
int interested[2]; // all values initially 0
void enter_region(int process)
{
    int other;
    other = 1 - process; // position 1
    interested[process]=1; // position 2
    turn = process; // position 3
    while(turn == process && interested[other] == 1);
}
void leave_region(int process)
{
    interested[process] = 0;
}
```

**Solution**

<table>
<thead>
<tr>
<th>Process 0</th>
<th>Process 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>other = 1</td>
<td>other = 0, interested[1] = 1, turn = 1</td>
</tr>
<tr>
<td>interested[0] = 1</td>
<td>check while conditions: turn==1(true) and interested[0]==1(true) then process 1 wait and give CPU back to process 0</td>
</tr>
<tr>
<td>turn = 0</td>
<td>stop, interrupt by process 1</td>
</tr>
<tr>
<td>stop, interrupt by process 1</td>
<td>check while condition turn==0(false) and interested[1]==1(true) then process 0 enter CS(critical section)</td>
</tr>
<tr>
<td>process 0 keeps working then finish CS job, after that interested[0]=0 and give CPU to process 1</td>
<td>check while conditions: turn==1(true) and interested[0]==1(true) then process 1 wait and give CPU back to process 0</td>
</tr>
</tbody>
</table>

**Q2** The Producer and Consumer problem is under consideration. Trace a semaphore based solution of the problem and write the output, if
• There is one customer and one producer
• Customer starts first
• Scheduler terminates a process whenever it enters to its critical region and another process gets turn

typedef int semaphore;

// shared variables
semaphore mutex = 1;     // controls access to critical region
semaphore empty = N;     // counts empty buffer slots
semaphore full = 0;      // counts full buffer slots

void producer(void)
{
    int item;
    while(1) {
        produce_item(&item);
        down(&empty);      // decrement empty count
        down(&mutex);      // enter critical region
        enter_item(&item); // put new item in buffer
        up(&mutex);        // leave critical region
        up(&full);         // increment count of full slots
    }
}

void consumer(void)
{
    int item;
    while(1) {
        down(&full);       // decrement full count
        down(&mutex);      // enter critical region
        remove_item(&item);// take item from buffer
        up(&mutex);        // leave critical region
        up(&empty);        // increment count of empty slots
        consume_item(item);
    }
}

Solution

<table>
<thead>
<tr>
<th>consumer</th>
<th>Producer</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop, interrupt by producer</td>
<td>produce_item(&amp;item)</td>
</tr>
<tr>
<td></td>
<td>down(&amp;empty), empty = N-1</td>
</tr>
<tr>
<td></td>
<td>down(&amp;mutex), mutex is now 0</td>
</tr>
<tr>
<td></td>
<td>stop, interrupt by consumer</td>
</tr>
<tr>
<td>stop, interrupt by producer</td>
<td>enter_item(&amp;item)</td>
</tr>
<tr>
<td></td>
<td>up(&amp;mutex), mutex = 1</td>
</tr>
<tr>
<td></td>
<td>up(&amp;full), full = 1</td>
</tr>
<tr>
<td></td>
<td>stop, interrupt by consumer</td>
</tr>
</tbody>
</table>
Q3 A computer has enough room to hold three processes in its main memory. 1st process spends 40% of its time for CPU utilization. This measure is 50% and 60% for 2nd and 3rd processes, respectively. Calculate overall wasted time (I/O wait time).

Solution

\[
\text{Overall}_{\text{wasted\_time}} = \left(1 - \frac{2}{5}\right) \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{3}{5}\right) = \frac{3}{5} \cdot \frac{1}{2} \cdot \frac{2}{5} = \frac{6}{50} = \frac{3}{25}.
\]

Q4. Consider a swapping system in which memory consists of the memory holes of 32KB, 26KB, 24KB, 20KB, and 30 KB. Which hole is taken for successive segment requests of 20 KB, 25KB and 30KB in first fit? Now repeat the question for next fit, best fit and worst fit.

Solution

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>32</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>NF</td>
<td>32</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>BF</td>
<td>20</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>WF</td>
<td>32</td>
<td>30</td>
<td>-</td>
</tr>
</tbody>
</table>