Lab 3
Process Scheduling

Objective
In this experiment, students will be exposed to:
- Different scheduling algorithms
- Problems that can arise from different scheduling algorithms
- Learn about process/task starvation, priority inversion, etc.

Prelab
In the prelab you should include the definitions for the following terms: shortest job first, first come first served, foreground-background, round robin, polled scheduling, priority based scheduling, priority inversion, and starvation. Furthermore, you should investigate RTAI functions to initialize and signal semaphores in Kernel Space: rt_sem_init, rt_sem_delete, rt_sem_wait, rt_sem_signal; and the following functions: rt_sleep and rt_busy_sleep.

Lab Procedure
The overall purpose of this lab is to implement a traffic light using the TS-7250 board and the auxiliary board provided in class. This auxiliary board contains three lights which represent the signals for one direction, the other direction, and pedestrians. When a light is turned on it represents a green light or WALK sign and when turned off it represents a red light. Also, the far left button on the auxiliary board represents the push button for the pedestrians to use when they wish to cross the street. The lights on the board are connected to the top three pins of Port B on the board. Be sure to only change the direction of these three pins when you use the lights.
Extra Module
For this lab, the pedestrian button will need to be checked to see if it has been pushed. This requires the functions:

```c
int check_button(void)  // this function returns the status of the button:
  // 1 if pressed

void clear_button(void)  // after you have handled a pushed button
  // event call this function
```

These functions are a part of the ece4220lab3.o module that needs to be installed before running your program. The module and its header file can be found in: /opt/ts7250/students.

Part 1: Polled Scheduling
For this part of the lab, you are to set up a single real time task that acts as a scheduler. It should turn on the light corresponding to one direction, then the light for the other direction, and then check if the pedestrian light needs to be turned on by checking the status of the button.

Questions:
- Can this implementation be considered real time? Why or why not?
- How can you improve the scheduling so the implementation acts more like a real time system?

Part 2: Priority Scheduling
For Part 2 you are to create three real time tasks, where each task is responsible for turning on and off one of the lights. Because the lights are a shared resource for all tasks, you need to implement some sort of protection to ensure that no more than one light can be on at any given time.

In your code, the timings of the traffic lights should be implemented with the rt sleep functions you investigated for the prelab. Be careful when/if you use `rt_busy_sleep()`...

For this section, you should experiment with the following combination of priority levels for each task and report the corresponding observations.

<table>
<thead>
<tr>
<th>Case No.</th>
<th>Priority Combinations</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PTL1 = PTL2 = PPL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PTL1 = PTL2 &gt; PPL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PTL1 = PTL2 &lt; PPL</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PTL1 &gt; PTL2 &gt; PPL</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTL2 &lt; PTL1 &lt; PPL</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PTL1 &lt; PTL2 = PPL</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PTL1 &gt; PTL2 = PPL</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** PTL1, PTL2 and PPL are the priorities of traffic light-1, traffic light-2 and the pedestrian light, respectively.

Which configuration(s) of priorities will lead to a round robin scheduling scheme? Which configuration(s) will cause a task to starve? What other configurations can you find? Make sure you report all your findings and explain/discuss the results thoroughly.

**Questions:**
- What are some problems that you ran into with this implementation?
- How did you fix your code to overcome these problems?
- What happens if the pedestrian button is pressed frequently/rapidly?
- What is the difference between `rt_busy_sleep()` and `rt_sleep()`? Did you use both of them? Did you experience any problems?

**Post Lab:**
Be sure your lab report contains answers to all of the questions asked in this lab plus the description of your modules/tasks/functions, goals, comments, results, conclusions, etc.

**Hints:**
1. Before installing any of your modules, remove the `rtai_fifos` module and the `rtai_lxrt` module.
2. Also, since we are using real time semaphores, you will need to install the semaphore module: `rtai_sem.o` located in `/usr/realtime/modules`. 