1) Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single-processor system? Elaborate please.

2) Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	5	4
P_2	3	1
P_3	1	2
P_4	7	2
P_5	4	3

The processes are assumed to have arrived in the order P1 , P2 , P3 , P4 , P5 , all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for each of these scheduling algorithms?
d. d. Which of the algorithms results in the minimum average waiting time (over all processes)?

3) Explain the how the following scheduling algorithms discriminate either in favor of or against short processes:

a. FCFS
b. RR
c. Multilevel feedback queues

4) Find an example for one of the three algorithms above that is being used in the industry nowadays. Explain the technology presented and analyze it based on you current knowledge of the OperatingSystems (from process, threading and scheduling algorithms standpoint)

Programming Projects:
P-29 of your textbook
Programs are accepted in C, C++, Python and Go programming Languages.

The project explanation is as follows:

Scheduling Algorithms!

This project involves implementing several different process scheduling algo -rithms. The scheduler will be assigned a predefined set of tasks and will schedule the tasks based on the selected scheduling algorithm. Each task is assigned a priority and CPU burst. The following scheduling algorithms will be implemented:

- First-come, first-served (FCFS), which schedules tasks in the order in which they request the CPU.
- Shortest-job-first (SJF), which schedules tasks in order of the length of the tasks' next CPU burst.
- Priority scheduling, which schedules tasks based on priority.
- Round-robin (RR) scheduling, where each task is run for a time quantum (or for the remainder of its CPU burst).
- Priority with round-robin, which schedules tasks in order of priority and uses round-robin scheduling for tasks with equal priority.

Priorities range from 1 to 10, where a higher numeric value indicates a higher relative priority. For round-robin scheduling, the length of a time quantum is 10 milliseconds.

Implementation:

The implementation of this project may be completed in either C or Java, and program files supporting both of these languages are provided in the source code download for the text. These supporting files read in the schedule of tasks, insert the tasks into a list, and invoke the scheduler.
The schedule of tasks has the form [task name] [priority] [CPU burst], with the following example format:

```
T1, 4, 20
T2, 2, 25
T3, 3, 25
T4, 3, 15
T5, 10, 10
```

Thus, task T1 has priority 4 and a CPU burst of 20 milliseconds, and so forth. It is assumed that all tasks arrive at the same time, so your scheduler algorithms do not have to support higher-priority processes preempting processes with lower priorities. In addition, tasks do not have to be placed into a queue or list in any particular order.
There are a few different strategies for organizing the list of tasks, as first presented in Section 5.1.2. One approach is to place all tasks in a single

unordered list, where the strategy for task selection depends on the scheduling algorithm. For example, SJF scheduling would search the list to find the task with the shortest next CPU burst. Alternatively, a list could be ordered according to scheduling criteria (that is, by priority). One other strategy involves having a separate queue for each unique priority, as shown in Figure 5.7. These approaches are briefly discussed in Section 5.3.6. It is also worth highlighting that we are using the terms list and queue somewhat interchangeably. However, a queue has very specific FIFO functionality, whereas a list does not have such strict insertion and deletion requirements. You are likely to find the functionality of a general list to be more suitable when completing this project.