Nitin Kumar
What we understood:

Priority Scheduling (Pre-emptive and Non Pre-emptive):
Each Process has a priority associated with it.
A Process with highest priority is selected for scheduling among all other processes in the ready queue. In case of Conflict a process with minimum arrival time is selected and if that also leads to conflict then a process with minimum process ID is selected.
Drawback of priority:
Priority Algorithm is unfair for I/O bound processes as they are given less priority.
Alternative:
Use of Completely fair Scheduler algorithm as it does elegant handling of I/O and CPU bound processes.

Varun Reddy

Shortest Job First (Non-Pre-emptive) (SJF):
Out of all the available processes during scheduling, the process with least burst time is considered and in case of a tie, one of the tied processes with lowest process ID is considered. Since this is a non-preemptive version, the process is run until completion or when I/O request is to be handled by the process.
Shortest Job First (Pre-emptive) or **Shortest Remaining Time First (SRTF):**
Out of all the available processes during scheduling, the process with least burst time is considered and in case of a tie, one of the tied processes with lowest process ID is considered. Since this is a preemptive version, if a process which has **strictly shorter remaining time** than present process comes into queue, the present running process is preempted and the newly arrived shorter process is scheduled.

Advantages:
- **Theoretically,** this is the best algorithm (optimal algorithm)
- It acts as benchmark for other scheduling algorithms when comparing performances

Drawbacks:
- Burst times of processes are not known beforehand and hence it is difficult to implement this algorithm practically.
- **SRTF** has more context switches and scheduling overheads than **SJF** as processes are more frequently preempted in **SRTF**

- This algorithm (both preemptive and non-preemptive versions) can cause starvation to processes with higher burst times.

Alternatives:

- We can predict the burst times of processes using various techniques
    - **Static techniques**
        - Using this we can predict the **total burst time**
        - It is predicted by using
            - **Size of process**
            - **Type of process**
                - Foreground processes are given more priority than background processes
    - **Dynamic techniques**
        - Using this we can predict the **next CPU burst**
        - It is predicted by using
            - **Simple averaging**
            - **Exponential averaging**

First Come First Serve Scheduling:
A process which arrives first is selected for scheduling among all other processes in the ready queue. In case of conflict, a process with the minimum process id is selected. In this way all the Processes are selected from the ready queue.
Drawback: Whenever there is a process with large burst time before the process with small burst time, the latter process can get CPU time only after the former process has finished. This property leads to a situation called Convoy Effect.

Round Robin Scheduling:
Each process is assigned a fixed time (Time Quantum/Time Slice) in a cyclic way**.**To implement Round Robin scheduling, we keep the ready queue as a FIFO queue of processes. The CPU scheduler picks the first process from the ready queue and executes it for 1-time quantum, and then changes the process. If the process burst time is less than 1-time quantum then it will implicitly gets the other process from the ready queue and removes it from the ready queue, else it will be added at the end of the queue to complete it execution.

Pratik Haware
What we understood:

Multi level queue:

1. Normally, we have a single ready queue for processes. When the processor is available, it selects a process from this queue
2. However, not all processes are equal. Interactive processes require quick turn around time or else the user will think that his computer is slow. On the other hand, batch processes usually run in the background and their turn around time is not very critical.
3. Hence, it makes sense to have separate ready queues for interactive and batch processes.
4. Higher priority should be given to interactive process queue. When this queue is empty, then we can start to schedule the batch process queue.
5. Round robin can be used for interactive process queue so that all processes get equal time slices and no process seems stuck for the user. First Come First Serve (FCFS) can be used for batch process queue because their turnaround time is not critical
6. We can classify the processes in as many categories as we want (System, Interactive, Batch, etc) and can have a ready queue for each category. The policy decides the priority for each queue.

What we will be doing:
1. We have to make a simulator for scheduling processes
2. We will be given an input.csv file which has characteristics of the processes
3. Our program should simulate the scheduling of the processes given in the input file and output some statistics.
4. **Priority Non-Preemptive & Priority Preemptive:** Min-Heap (or Max-Heap) which internally Red-Black Tree is used to find the process with highest Priority, in order to schedule the process.
First Come First Serve: Sort the given processes according to the arrival time and then execute each of the processes.
Round Robin: Maintain two queues, one for checking the completion of process and other queue to maintain the order of execution of process.

Shortest job first and Shortest Remaining Time first: Min-Heap (which internally uses Red-Black Tree) is used to find the process with lowest burst time, in order to schedule the process.

5. All I/O operations can run in parallel.
6. Statistics will be generated based on the scheduling algorithms.
7. These statistics can be visualized in the form of a graph, pie chart, etc with the help of python.
8. By changing some parameters (Eg: time slice in round robin), we will get different statistics. By visualizing them, we can get an idea how choosing different scheduling algorithms affect the amount of resources allocated to processes
9. Even with same algorithms, changing their parameters (Eg: time slice in round robin) greatly affect the resources allocated to processes

What we would be doing in the project:

1. We have to implement 8 scheduling algorithms. Each person has taken responsibility for 2 scheduling algorithms.
2. As of now, each person has implemented 2 algorithms.