Q2)

Program Logic: A) counter: counter initiates an unsigned int as 1 then starts increasing it by 1 in a while loop until it overflows and reaches its maximum value (2^32)

B)Launch three instances of counter with the specific parameters and scheduling policies defined separately(sched_fifo,rr,other are launched which respectively launch counter with the policies)

C) in a for loop fork three different children and note the PIDs of each as well as a start timestamp . wait for a process to complete using the wait(&status) clause compare the pid returned by wait(with the pids of the children spawned earlier) to see which process has ended and save its end timestamp (this ensures that processes are not mixed up amongst each other and wait statements are independent of order assumptions)

Execute a python script at the end with the three values for fifo rr other as stdin arguments and plot a histogram using matplotlib

OUTCOMES: the script reproducibly returns three different timestamps, with minor differences, in the scale of ~2 seconds and the order tends to be FIFO<RR<OTHER.

This can be explained by the precedence of the different policies provided by the linux scheduler

Q3)

Program Logic: Count Running Processes Kernel Module

Objective: The Kernel Module is designed to count and display information about running processes in a Linux system using the task_struct data structure provided by the Linux kernel.

The program is compatible with the kernel of UBUNTU VERSION 18.

Functionality:

1. **Number of Currently Running Processes:**
    - **The module iterates through all processes using for_each_process and checks the state of each process.**
    - **If the state of a process is TASK_RUNNING, it increments a counter to count running processes.**
    - **The module then prints the total number of running processes to the kernel log.**
2. **Number of Interruptible Processes:**
    - **Similar to the above process, the module iterates through all processes.**
    - **If the state of a process is either TASK_RUNNING or TASK_INTERRUPTIBLE, it increments a counter.**
    - **The module prints the total number of interruptible processes to the kernel log.**
3. **Number of All Processes:**
    - **The module counts all processes in the system by iterating through each process.**
    - **It increments a counter for each process encountered.**
    - **The module prints the total number of all processes in the system to the kernel log.**

Usage:

1. **Build the Module:**
    - **Use the provided Makefile to build the kernel module using the make command.**
2. **Load the Module:**
    - **Load the kernel module into the kernel using sudo insmod q.ko.**
3. **Check Kernel Logs:**
    - **View the kernel logs using dmesg to see the results, including the counts of running, interruptible, and all processes.**

4. **Unload the Module:**
    - **Remove the kernel module when done using sudo rmmod count_processes.**