Name : Hanuman Bavane  
Roll No :13109  
Batch : A1  
  
 Practical 5

def fcfs\_scheduling():

    processes = int(input("Enter the number of processes (up to 6): "))

    bt = [0] \* processes  # Burst times

    wt = [0] \* processes  # Waiting times

    tt = [0] \* processes  # Turnaround times

    arrival\_times = list(range(processes))  # Arrival times: 0, 1, 2, ..., processes-1

    print("Enter burst times:")

    for i in range(processes):

        bt[i] = int(input(f"Process {i + 1}: "))

    exit\_times = [0] \* processes

    for i in range(processes):

        if i == 0:

            exit\_times[i] = arrival\_times[i] + bt[i]

        else:

            exit\_times[i] = max(arrival\_times[i], exit\_times[i - 1]) + bt[i]

        tt[i] = exit\_times[i] - arrival\_times[i]

        wt[i] = tt[i] - bt[i]

    total\_wt = sum(wt)

    total\_tt = sum(tt)

    print("\nProcess\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time")

    for i in range(processes):

        print(f"{i + 1:<7}\t{bt[i]:<12}\t{arrival\_times[i]:<14}\t{wt[i]:<14}\t{tt[i]:<15}")

    print(f"\nAverage Waiting Time: {total\_wt / processes:.2f}")

    print(f"Average Turnaround Time: {total\_tt / processes:.2f}")

def priority\_scheduling():

    totalprocess = int(input("Enter the number of processes: "))

    proc = [[0, 0, 0, 0] for \_ in range(totalprocess)]

    def get\_wt\_time(wt):

        service = [0] \* totalprocess

        service[0] = proc[0][0]

        wt[0] = 0

        for i in range(1, totalprocess):

            service[i] = service[i - 1] + proc[i - 1][1]

            wt[i] = service[i] - proc[i][0]

            if wt[i] < 0:

                wt[i] = 0

    def get\_tat\_time(tat, wt):

        for i in range(totalprocess):

            tat[i] = proc[i][1] + wt[i]

    def findgc():

        wt = [0] \* totalprocess

        tat = [0] \* totalprocess

        wavg = 0

        tavg = 0

        get\_wt\_time(wt)

        get\_tat\_time(tat, wt)

        stime = [0] \* totalprocess

        ctime = [0] \* totalprocess

        stime[0] = proc[0][0]

        ctime[0] = stime[0] + proc[0][1]

        for i in range(1, totalprocess):

            stime[i] = max(proc[i][0], ctime[i - 1])

            ctime[i] = stime[i] + proc[i][1]

        print("Process\tArrival\tPriority\tBurst\tStart\tComplete\tTurnaround\tWaiting")

        for i in range(totalprocess):

            wavg += wt[i]

            tavg += tat[i]

            print(f"P{proc[i][3]}\t{proc[i][0]}\t{proc[i][2]}\t\t{proc[i][1]}\t{stime[i]}\t{ctime[i]}\t\t{tat[i]}\t\t{wt[i]}")

        print(f"\nAverage Waiting Time: {wavg / totalprocess:.2f}")

        print(f"Average Turnaround Time: {tavg / totalprocess:.2f}")

    for i in range(totalprocess):

        arrivaltime = int(input(f"Enter arrival time for process {i + 1}: "))

        bursttime = int(input(f"Enter burst time for process {i + 1}: "))

        priority = int(input(f"Enter priority for process {i + 1}: "))

        proc[i][0] = arrivaltime

        proc[i][1] = bursttime

        proc[i][2] = priority

        proc[i][3] = i + 1

    proc.sort(key=lambda x: (x[0], x[2]))

    findgc()

def round\_robin\_scheduling():

    processes = int(input("Enter the number of processes: "))

    quantum = int(input("Enter the quantum time: "))

    at = [0] \* processes

    bt = [0] \* processes

    rem\_bt = [0] \* processes

    print("Enter the arrival times:")

    for i in range(processes):

        at[i] = int(input(f"Process {i + 1} arrival time: "))

    print("Enter the burst times:")

    for i in range(processes):

        bt[i] = int(input(f"Process {i + 1} burst time: "))

        rem\_bt[i] = bt[i]

    wt = [0] \* processes

    ct = [0] \* processes

    tt = [0] \* processes

    t = 0

    complete = 0

    ready\_queue = []

    arrived = [False] \* processes

    while complete < processes:

        for i in range(processes):

            if at[i] <= t and not arrived[i]:

                ready\_queue.append(i)

                arrived[i] = True

        if ready\_queue:

            i = ready\_queue.pop(0)

            if rem\_bt[i] > quantum:

                t += quantum

                rem\_bt[i] -= quantum

                for j in range(processes):

                    if at[j] <= t and not arrived[j]:

                        ready\_queue.append(j)

                        arrived[j] = True

                ready\_queue.append(i)

            else:

                t += rem\_bt[i]

                rem\_bt[i] = 0

                ct[i] = t

                tt[i] = ct[i] - at[i]

                wt[i] = tt[i] - bt[i]

                complete += 1

        else:

            t += 1

    print("\nProcess\tArrival\tBurst\tCompletion\tWaiting\tTurnaround")

    for i in range(processes):

        print(f"P{i + 1}\t\t{at[i]}\t\t{bt[i]}\t\t{ct[i]}\t\t\t{wt[i]}\t\t{tt[i]}")

    total\_wt = sum(wt)

    total\_tt = sum(tt)

    print(f"\nAverage Waiting Time: {total\_wt / processes:.2f}")

    print(f"Average Turnaround Time: {total\_tt / processes:.2f}")

def sjf\_preemptive\_scheduling():

    processes = int(input("Enter the number of processes: "))

    arrival\_times = [0] \* processes

    burst\_times = [0] \* processes

    print("Enter the arrival times:")

    for i in range(processes):

        arrival\_times[i] = int(input(f"Process {i + 1}: "))

    print("Enter the burst times:")

    for i in range(processes):

        burst\_times[i] = int(input(f"Process {i + 1}: "))

    remaining\_times = burst\_times[:]

    completion\_times = [0] \* processes

    waiting\_times = [0] \* processes

    turnaround\_times = [0] \* processes

    time = 0

    completed = 0

    while completed < processes:

        min\_remaining\_time = float('inf')

        shortest\_process = None

        for i in range(processes):

            if arrival\_times[i] <= time and remaining\_times[i] > 0:

                if remaining\_times[i] < min\_remaining\_time:

                    min\_remaining\_time = remaining\_times[i]

                    shortest\_process = i

        if shortest\_process is None:

            time += 1

            continue

        remaining\_times[shortest\_process] -= 1

        if remaining\_times[shortest\_process] == 0:

            completed += 1

            completion\_times[shortest\_process] = time + 1

            waiting\_times[shortest\_process] = (completion\_times[shortest\_process] -

                                               arrival\_times[shortest\_process] -

                                               burst\_times[shortest\_process])

            turnaround\_times[shortest\_process] = (waiting\_times[shortest\_process] +

                                                  burst\_times[shortest\_process])

        time += 1

    total\_wt = sum(waiting\_times)

    total\_tt = sum(turnaround\_times)

    print("\nProcess\tArrival\tBurst\tCompletion\tWaiting\tTurnaround")

    for i in range(processes):

        print(f"P{i + 1}\t\t{arrival\_times[i]}\t\t{burst\_times[i]}\t\t{completion\_times[i]}\t\t{waiting\_times[i]}\t\t{turnaround\_times[i]}")

    print(f"\nAverage Waiting Time: {total\_wt / processes:.2f}")

    print(f"Average Turnaround Time: {total\_tt / processes:.2f}")

def main():

    while True:

        print("\nSelect Scheduling Algorithm:")

        print("1. FCFS")

        print("2. Priority Scheduling")

        print("3. Round Robin")

        print("4. SJF Preemptive")

        print("5. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:

            fcfs\_scheduling()

        elif choice == 2:

            priority\_scheduling()

        elif choice == 3:

            round\_robin\_scheduling()

        elif choice == 4:

            sjf\_preemptive\_scheduling()

        elif choice == 5:

            print("Exiting the program.")

            break  # Exit the loop

        else:

            print("Invalid choice! Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

OUTPUT :

PS C:\Users\HP> & "C:/Program Files/Python312/python.exe" "c:/Users/HP/OneDrive/Desktop/spos practical/lexical/fifo.py"

Select Scheduling Algorithm:

1. FCFS

2. Priority Scheduling

3. Round Robin

4. SJF Preemptive

5. Exit

Enter your choice: 1

Enter the number of processes (up to 6): 5

Enter burst times:

Process 1: 4

Process 2: 3

Process 3: 1

Process 4: 2

Process 5: 5

Process Burst Time Arrival Time Waiting Time Turnaround Time

1 4 0 0 4

2 3 1 3 6

3 1 2 5 6

4 2 3 5 7

5 5 4 6 11

Average Waiting Time: 3.80

Average Turnaround Time: 6.80

Select Scheduling Algorithm:

1. FCFS

2. Priority Scheduling

3. Round Robin

4. SJF Preemptive

5. Exit

Enter your choice: 2

Enter the number of processes: 3

Enter arrival time for process 1: 0

Enter burst time for process 1: 5

Enter priority for process 1: 2

Enter arrival time for process 2: 1

Enter burst time for process 2: 3

Enter priority for process 2: 1

Enter arrival time for process 3: 2

Enter burst time for process 3: 8

Enter priority for process 3: 3

Process Arrival Priority Burst Start Complete Turnaround Waiting

P1 0 2 5 0 5 5 0

P2 1 1 3 5 8 7 4

P3 2 3 8 8 16 14 6

Average Waiting Time: 3.33

Average Turnaround Time: 8.67

Select Scheduling Algorithm:

1. FCFS

2. Priority Scheduling

3. Round Robin

4. SJF Preemptive

5. Exit

Enter your choice: 3

Enter the number of processes: 4

Enter the quantum time: 2

Enter the arrival times:

Process 1 arrival time: 0

Process 2 arrival time: 1

Process 3 arrival time: 2

Process 4 arrival time: 4

Enter the burst times:

Process 1 burst time: 5

Process 2 burst time: 4

Process 3 burst time: 2

Process 4 burst time: 1

Process Arrival Burst Completion Waiting Turnaround

P1 0 5 12 7 12

P2 1 4 11 6 10

P3 2 2 6 2 4

P4 4 1 9 4 5

Average Waiting Time: 4.75

Average Turnaround Time: 7.75

Select Scheduling Algorithm:

1. FCFS

2. Priority Scheduling

3. Round Robin

4. SJF Preemptive

5. Exit

Enter your choice: 4

Enter the number of processes: 4

Enter the arrival times:

Process 1: 1

Process 2: 2

Process 3: 1

Process 4: 4

Enter the burst times:

Process 1: 3

Process 2: 4

Process 3: 2

Process 4: 4

Process Arrival Burst Completion Waiting Turnaround

P1 1 3 6 2 5

P2 2 4 10 4 8

P3 1 2 3 0 2

P4 4 4 14 6 10

Average Waiting Time: 3.00

Average Turnaround Time: 6.25

Select Scheduling Algorithm:

1. FCFS

2. Priority Scheduling

3. Round Robin

4. SJF Preemptive

5. Exit

Enter your choice: 5

Exiting the program.

PS C:\Users\HP\Desktop\New folder (2)>

Conclusion :

The tested scheduling algorithms included:

1. **FCFS**: Processes executed in arrival order.
2. **Priority Scheduling**: Higher-priority processes executed first.
3. **Round Robin**: Processes shared CPU time with a quantum of **2**.
4. **SJF Preemptive**: Prioritized processes with the shortest burst times.

Overall, **SJF Preemptive** performed best in managing process execution.