**UIT2512---Operating Systems Practices Lab**

# 2) Implementation of Round Robin CPU Scheduling Algorithm in Python

# Name: Vasundhara.B

# Roll no: 3122 21 5002 119

# AIM:

# To implement the Round Robin CPU Scheduling Algorithm in Python, illustrating the allocation of CPU time to processes in a circular fashion with a fixed time quantum.

# Description:

This algorithm employs a fixed time quantum, allowing each process to execute for a specified time slice before moving on to the next process. This implementation in Python will simulate the round-robin scheduling of processes, showcasing how it ensures fairness among processes by preventing any single process from monopolizing the CPU for extended periods.

**CODE:**

def findWaitingTime(processes, n, bt, wt, quantum):

    rem\_bt = [0] \* n

    for i in range(n):

        rem\_bt[i] = bt[i]

    t = 0  # Current time

    gantt\_chart = []  # List to store the Gantt chart

    while True:

        done = True

        for i in range(n):

            if rem\_bt[i] > 0:

                done = False  # There is a pending process

                if rem\_bt[i] > quantum:

                    t += quantum

                    rem\_bt[i] -= quantum

                else:

                    t += rem\_bt[i]

                    wt[i] = t - bt[i]

                    rem\_bt[i] = 0

                # Add the process to the Gantt chart

                gantt\_chart.append(processes[i])

        if done:

            break

    return gantt\_chart

def findTurnAroundTime(processes, n, bt, wt, tat):

    for i in range(n):

        tat[i] = bt[i] + wt[i]

def findavgTime(processes, n, bt, quantum):

    wt = [0] \* n

    tat = [0] \* n

    gantt\_chart = findWaitingTime(processes, n, bt, wt, quantum)

    findTurnAroundTime(processes, n, bt, wt, tat)

    print()

    print("Processes Burst Time Waiting",

          "Time Turn-Around Time")

    total\_wt = 0

    total\_tat = 0

    for i in range(n):

        total\_wt = total\_wt + wt[i]

        total\_tat = total\_tat + tat[i]

        print("-----------------------------------------------------------")

        print(" ", processes[i], "\t\t", bt[i],

              "\t\t", wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.5f " % (total\_wt / n))

    print("Average turn around time = %.5f " % (total\_tat / n))

    # Print Gantt Chart

    print("\nGantt Chart:")

    print("-" \* 40)

    prev\_process = gantt\_chart[0]

    for process in gantt\_chart:

        print("|", process, end=" ")

    print()

    print("-" \* 40)

# Driver code

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

    # Process id's

    proc = []

    burst\_time=[]

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

    n = int(input("Enter the no.of processes: "))

    for i in range(n):

        proc.append(i+1)

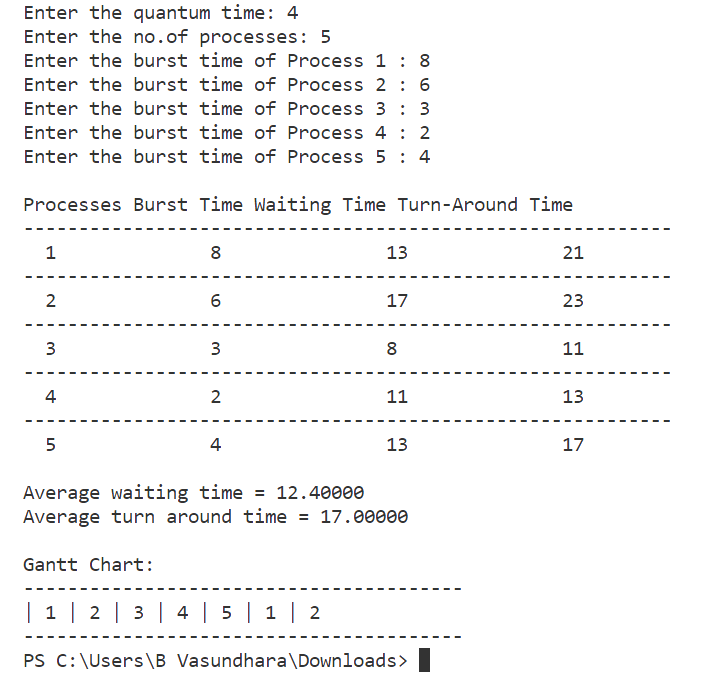
        bt=int(input(f"Enter the burst time of Process {i+1} : "))

        burst\_time.append(bt)

    # Time quantum

    findavgTime(proc, n, burst\_time, quantum)

**OUTPUT:**

****