Student Name: VIVEK KUMAR

Student ID: 11803833 **Roll Number:** 49 **Section:** K18HV

E mail: ranuyadav7781@gmail.com

GitHub Id: https://github.com/Vivek-kumar99/symmetrical-bassoon.git

QUESTION NUMBER 5

CPU schedules N processes which arrive at different time intervals and each process is allocated the CPU for a specific user input time unit, processes are scheduled using a preemptive round robin scheduling algorithm. Each process must be assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes one task has priority 0. The length of a time quantum is T units, where T is the custom time considered as time quantum for processing. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue. Design a scheduler so that the task with priority 0 does not starve for resources and gets the CPU at some time unit to execute. Also compute waiting time, turn around.

DISCRIPTION OF PROBLEM

CPU schedules N processes which arrive at different time intervals and each process allocated the CPU for a specific user input time unit, processes are scheduled using a preemptive round robin scheduling algorithm. Each process must be assigned a numerical priority, with a higher number indicating a higher relative priority. In addition, priority 0. The length of a time quantum time quantum for processing.

If is T units, where T is a process is to the processes one task has the custom time considered as preempted by a higher preempted process is placed at the end of t priority process, the he queue. Design a scheduler so that the task with priority 0 does not starve for resources and gets the CPU at some compute waiting time, turn around.

EXAMPLE:

Consider the set of 6 processes whose arrival time and burst time are given below-

PROCESS ID	ARRIVAL TIME	BURST TIME
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average turnaround time.

ANSWER:

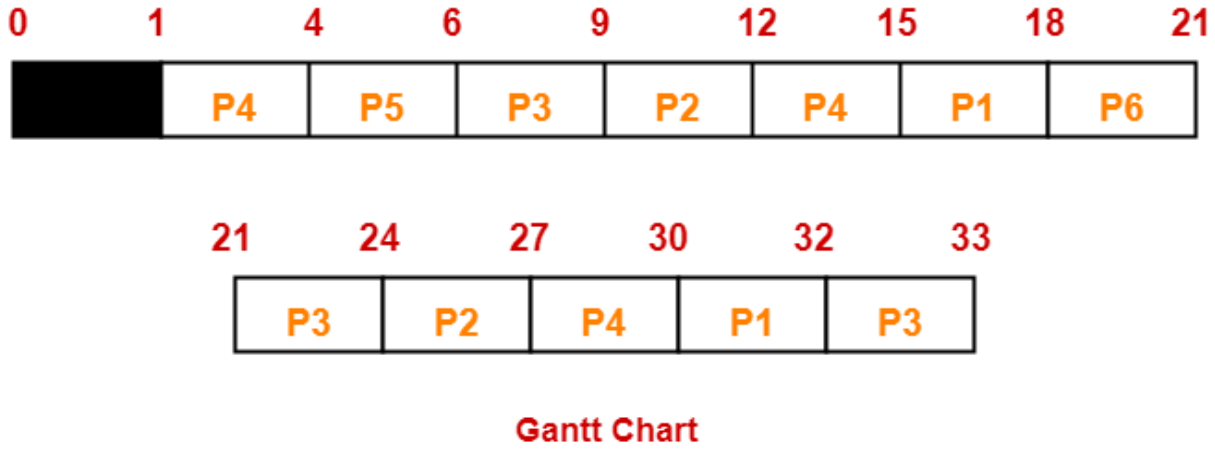
- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

PROCESS ID	EXIT TIME	TURN AROUND TIME	WAITING TIME
P1	32	32 – 5 = 27	27 – 5 = 22
P2	27	27 – 4 = 23	23 – 6 = 17
P3	33	33 – 3 = 30	30 – 7 = 23
P4	30	30 – 1 = 29	29 – 9 = 20
P5	6	6 – 2 = 4	4 – 2 = 2
P6	21	21 – 6 = 15	15 – 3 = 12

Average Turn Around time = (27 + 23 + 30 + 29 + 4 + 15) / 6 = 128 / 6 = 21.33 unit
Average waiting time = (22 + 17 + 23 + 20 + 2 + 12) / 6 = 96 / 6 = 16 unit

READY QUEUE:

P3, P1, P4, P2, P3, P6, P1, P4, P2, P3, P5, P4



Gantt Chart

PROGRAM CODE:

```
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,
        int bt[], int wt[], int quantum)
{
    int rem_bt[n];
    for(int i=0; i<n; i++)
        rem_bt[i] = bt[i];
    int t = 0;
    while(1)
    {
        bool done = true;
        for(int i = 0 ; i < n ; i++)
        {
            if(rem_bt[i] > 0)
            {
                done = false;
                if(rem_bt[i] > quantum)
                {
                  t += quantum;
                  rem_bt[i] -= quantum;
                }
                else
                {
                    t = t + rem_bt[i];
                    wt[i] = t - bt[i];
                    rem_bt[i] = 0;
```

```
                        }
                    }
                }
                if(done == true)
                break;
        }
}
void findTurnAroundTime(int processes[], int n,
                int bt[], int wt[], int tat[])
{
        for( int i=0; i<n; i++)
        tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n,int bt[],
                        int quantum)
{
        int wt[n], tat[n], total_wt=0, total_tat=0;
        findWaitingTime(processes, n, bt, wt, quantum);
        findTurnAroundTime(processes, n, bt, wt, tat);
        cout<< "Processes " << "  BurstTime"
            << "  WaitingTime"<< "  TurnAroundTime\n";
        for(int i=0; i<n; i++)
        {
                total_wt = total_wt + wt[i];
                total_tat = total_tat + tat[i];
                cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
         << wt[i] <<"\t\t " << tat[i] <<endl;
        }
```

```
    cout << "Average waiting time = "
            << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_tat / (float)n;
}
int main()
{
        int processes[] = {1,2,3};
        int n = sizeof processes / sizeof processes[0];
        int burst_time[] = {5,10,20};
        int quantum = 5;
        findavgTime(processes,n,burst_time,quantum);
        return 0;
}
```

ABOUT CODE:

1. void findWaitingTime(int processes[], int n,
int bt[], int wt[], int quantum)

Function to find the waiting time for all processes

2. void findTurnAroundTime(int processes[], int n,
int bt[], int wt[], int tat[])

Function to calculate turn around time

3.void findavgTime(int processes[], int n, int bt[],
int quantum)

Function to calculate average time

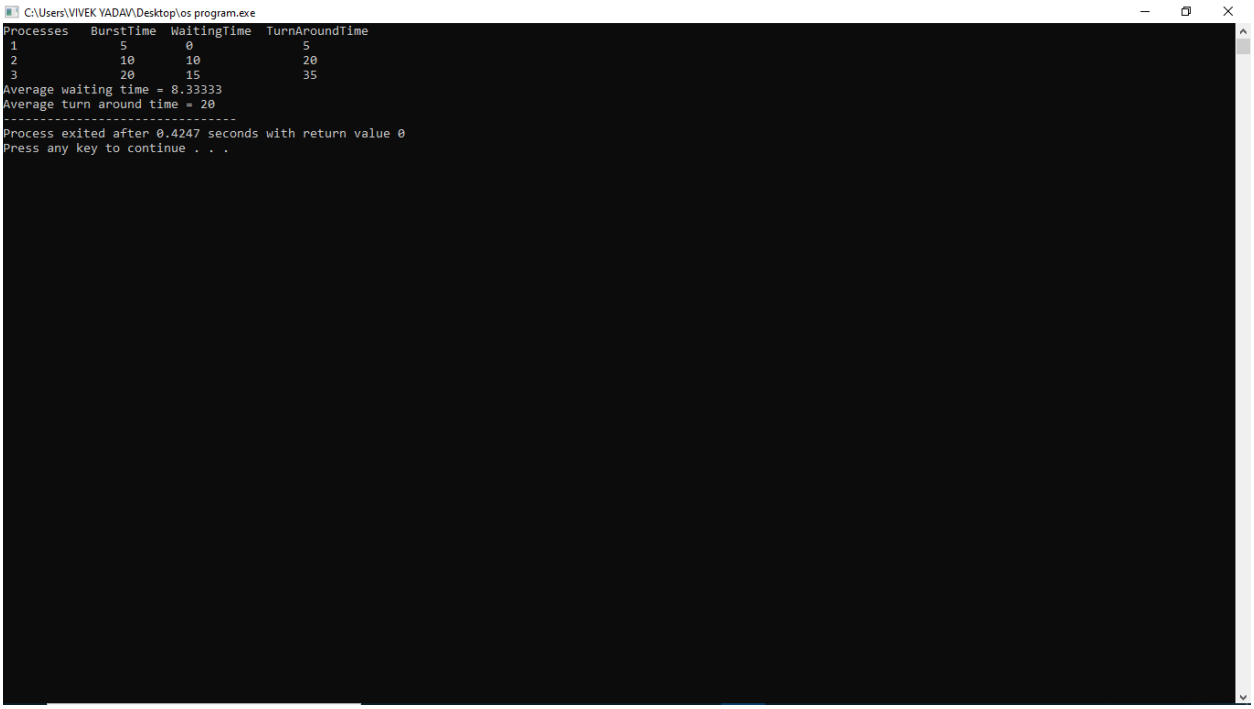
4. findWaitingTime(processes, n, bt, wt, quantum);

Function to find waiting time of all processes

5. findTurnAroundTime(processes, n, bt, wt, tat);

Function to find turn around time for all processes

OUTPUT:


C:\Users\VIVEK YADAV\Desktop\os program.exe
Processes BurstTime WaitingTime TurnAroundTime
1 5 0 5
2 10 10 20
3 20 15 35
Average waiting time = 8.33333
Average turn around time = 20
Process exited after 0.4247 seconds with return value 0
Press any key to continue . . .