**OS lab assignment 6**

**Agnim Gupta**

**2028082**

**A-23 CSSE**

**Question 1**

//Implementing pre emptive SJF

#include<stdio.h>

 int main()

{

    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

    float avg\_wt,avg\_tat;

    printf("Enter number of process:");

    scanf("%d",&n);

    printf("Enter Burst Time: ");

    for(i=0;i<n;i++)

    {

        printf("p%d:",i+1);

        scanf("%d",&bt[i]);

        p[i]=i+1;

    }

    for(i=0;i<n;i++)

    {

        pos=i;

        for(j=i+1;j<n;j++)

        {

            if(bt[j]<bt[pos])

                pos=j;

        }

        temp=bt[i];

        bt[i]=bt[pos];

        bt[pos]=temp;

        temp=p[i];

        p[i]=p[pos];

        p[pos]=temp;

    }

    wt[0]=0;

    for(i=1;i<n;i++)

    {

        wt[i]=0;

        for(j=0;j<i;j++)

            wt[i]+=bt[j];

        total+=wt[i];

    }

    avg\_wt=(float)total/n;

    total=0;

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time");

    for(i=0;i<n;i++)

    {

        tat[i]=bt[i]+wt[i];

        total+=tat[i];

        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

    }

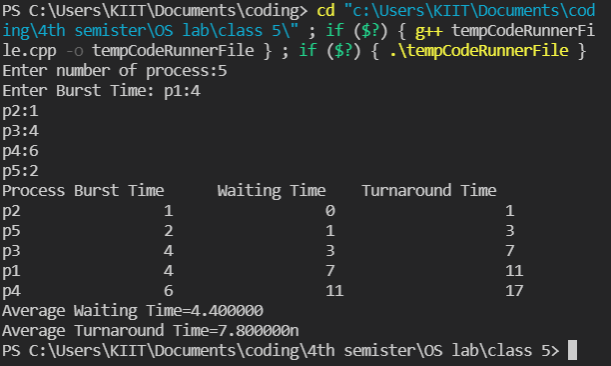
    avg\_tat=(float)total/n;

    printf("\nAverage Waiting Time=%f",avg\_wt);

    printf("\nAverage Turnaround Time=%fn",avg\_tat);

}

**Output**

****

**Question 2**

//RR scheduling

#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)

                {

                    //printf("I: %d\n",i);

                    t += quantum;

                    //printf("1: %d\n", t);

                    rem\_bt[i] -= quantum;

                    //printf("2: %d\n", rem\_bt[i]);

                }

                else

                {

                    //printf("R: %d\n", rem\_bt[i]);

                    t = t + rem\_bt[i];

                    //printf("value: %d\n", t);

                    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 "<< " Burst time "

        << " Waiting time " << " Turn around time\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 num,val;

    printf("enter number of processes:");

    scanf("%d",&num);

    int processes[num];

    printf("enter processes:");

    for (int i=0; i<num; i++)

    {

        scanf("%d",&val);

        processes[i]=val;

    }

    int n = sizeof processes / sizeof processes[0];

    int burst\_time[] = {3,4,3};

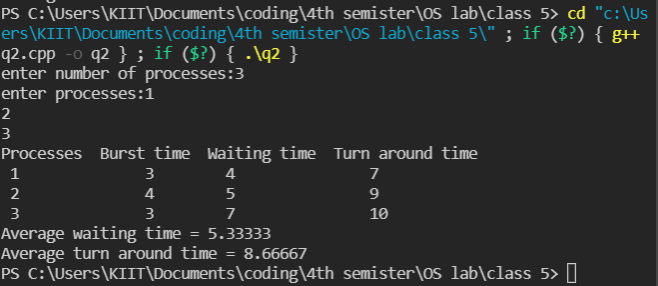
    int quantum = 2;

    findavgTime(processes, n, burst\_time, 2);

    return 0;

}

**Output**

****