**LAB – 5**

**AIM – Process Scheduling algorithm (FCFS, SJF)**

SHRUTI MISHRA

21BCP110

**FCFS** – First Come, First Served

**CODE:**

#include <stdio.h>

int main()

{

    int pid[15];

    int bt[15];

    int n;

    printf("Enter the number of processes: ");

    scanf("%d",&n);

    printf("Enter process id of all the processes: ");

    for(int i=0;i<n;i++)

    {

        scanf("%d",&pid[i]);

    }

    printf("Enter burst time of all the processes: ");

    for(int i=0;i<n;i++)

    {

        scanf("%d",&bt[i]);

    }

    int i, wt[n];

    wt[0]=0;

    //for calculating waiting time of each process

    for(i=1; i<n; i++)

    {

        wt[i]= bt[i-1]+ wt[i-1];

    }

    printf("Process ID     Burst Time     Waiting Time     TurnAround Time\n");

    float twt=0.0;

    float tat= 0.0;

    for(i=0; i<n; i++)

    {

        printf("%d\t\t", pid[i]);

        printf("%d\t\t", bt[i]);

        printf("%d\t\t", wt[i]);

        //calculating and printing turnaround time of each process

        printf("%d\t\t", bt[i]+wt[i]);

        printf("\n");

        //for calculating total waiting time

        twt += wt[i];

        //for calculating total turnaround time

        tat += (wt[i]+bt[i]);

    }

    float att,awt;

    //for calculating average waiting time

    awt = twt/n;

    //for calculating average turnaround time

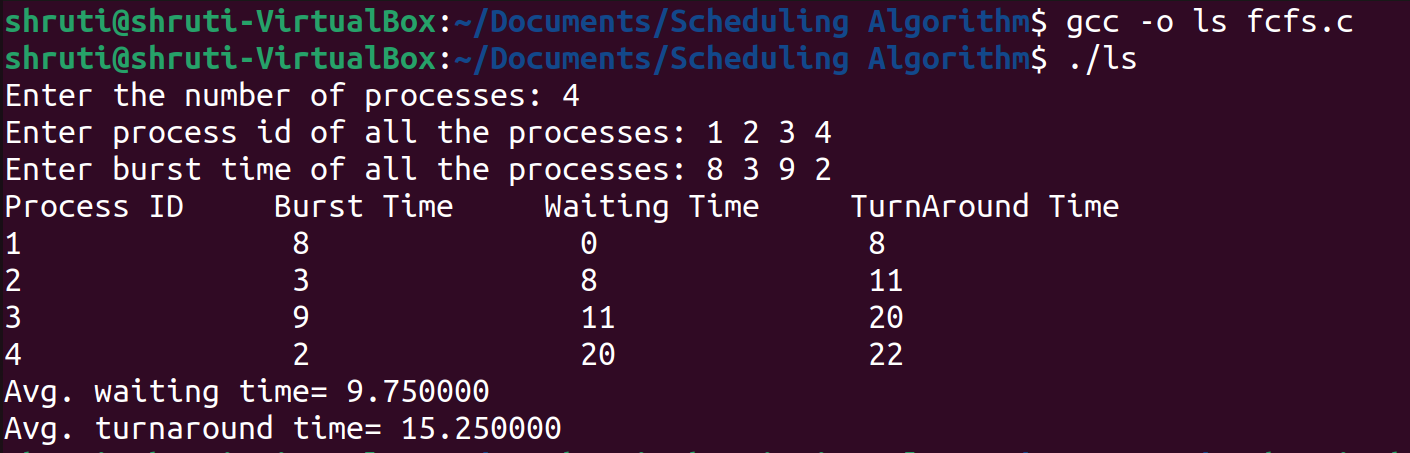
    att = tat/n;

    printf("Avg. waiting time= %f\n",awt);

    printf("Avg. turnaround time= %f\n",att);

}

**OUTPUT**:



**SJF** – Shortest job first

**CODE**

#include<stdio.h>

int main()

{

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

    float avg\_wt,avg\_tat;

    printf("Enter number of process:");

    scanf("%d",&n);

    printf("\nEnter Burst Time:\n");

    for(i=0;i<n;i++)

    {

        printf("p%d:",i+1);

        scanf("%d",&bt[i]);

        p[i]=i+1;

    }

    //sorting of burst times

    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;

    //finding the waiting time of all the processes

    for(i=1;i<n;i++)

    {

        wt[i]=0;

        for(j=0;j<i;j++)

             //individual WT by adding BT of all previous completed processes

            wt[i]+=bt[j];

        //total waiting time

        total+=wt[i];

    }

    //average waiting time

    avg\_wt=(float)total/n;

    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

    for(i=0;i<n;i++)

    {

        //turnaround time of individual processes

        tat[i]=bt[i]+wt[i];

        //total turnaround time

        totalT+=tat[i];

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

    }

    //average turnaround time

    avg\_tat=(float)totalT/n;

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

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

}

**OUTPUT**:

