**EXERCISE – 2**

**Name – Bhavy Kharbanda**

**SAP ID – 500082531**

**Roll Number – R214220342**

**Batch – B11**

**Date of Performance – 15/03/2021**

**Date of Submission – 27/03/2021**

OPERATING SYSTEM LAB

EXERCISE – 2.1

FIRST COME FIRST SERVE

**AIM:**  To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE.

**THEORY:**

CPU scheduler will decide which process should be given the CPU for its execution. For this, it use different algorithm to choose among the process. One among that algorithm is FCFS algorithm. In this algorithm the process which arrives first is given the CPU after finishing its request only it will allow CPU to execute other process.

**ALGORITHM:-**

**Step1: Create the number of process.**

**Step2: Get the ID and Service time for each process.**

**Step3: Initially, Waiting time of first process is zero and Total time for the first process is the starting time of that process. Step4: Calculate the Total time and Processing time for the remaining processes.**

**Step5: Waiting time of one process is the Total time of the previous process.**

**Step6: Total time of process is calculated by adding Waiting time and Service time.**

**Step7: Total waiting time is calculated by adding the waiting time for lack process.**

**Step8: Total turnaround time is calculated by adding all total time of each process.**

**Step9: Calculate Average waiting time by dividing the total waiting time by total number of process.**

**Step10: Calculate Average turnaround time by dividing the total time by the number of process.**

**Step11: Display the result.**

**Code:-**

#include<stdio.h>

int main()

{

int n, bt[20], wt[20], tat[20],i,j;

float avwt=0, avtat=0;

//bt->Burst time

//wt->Waiting time

//tat->Turn around time

//avwt-> Average waiting time

//avtat-> average turn around time

printf("\n Enter total number of processes:");

scanf("%d", &n);

printf("\n Enter process burst time:\n");

for(i=0;i<n;i++)

{

printf("P[%d]:", i+1);

scanf("%d", &bt[i]);

}

wt[0]=0; // Initial waiting time is ZERO

for(i=0;i<n;i++)

{

wt[i]=0;

for(j=0;j<i;j++)

wt[i]+=bt[j];

}

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

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i];

avwt+=wt[i];

avtat+=tat[i];

printf("\n P[%d]\t\t%d\t\t%d\t\t%d", i+1,bt[i],wt[i],tat[i]);

}

avwt/=i;

avtat/=i;

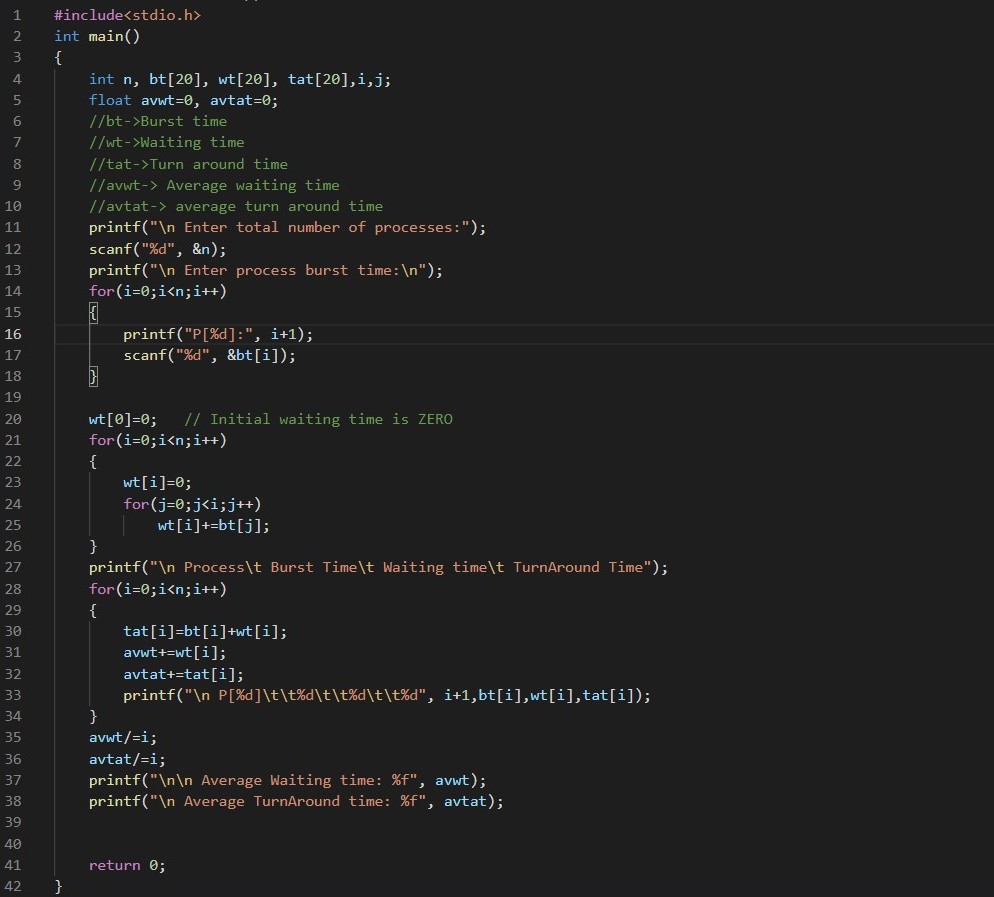
printf("\n\n Average Waiting time: %f", avwt);

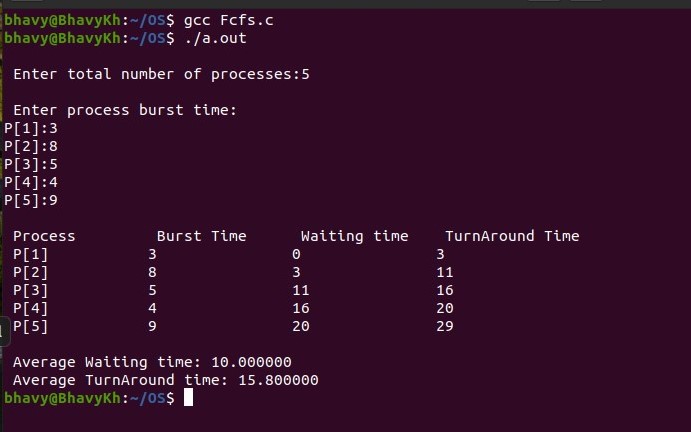
printf("\n Average TurnAround time: %f", avtat);

return 0;

}

**Screenshot/Output:-**

****

****

**\*\*\*\*\*\*\*End of Exercise - 2.1\*\*\*\*\*\*\***

OPERATING SYSTEM LAB

EXERCISE – 2.2

SHORTEST JOB FIRST

**AIM:** To write a C program to implement the CPU scheduling algorithm for shortest job first.

**THEORY:**

CPU scheduler will decide which process should be given the CPU for its execution. For this it use different algorithm to choose among the process. One among that algorithm is SJF algorithm. In this algorithm the process which has less service time given the CPU after finishing its request only it will allow CPU to execute next other process

**Algorithm:-**

**Step1: Get the number of process.**

**Step2: Get the id and service time for each process.**

**Step3: Initially the waiting time of first short process as 0 and total time of first short is process the service time of that process. Step4: Calculate the total time and waiting time of remaining process.**

**Step5: Waiting time of one process is the total time of the previous process.**

**Step6: Total time of process is calculated by adding the waiting time and service time of each process.**

**Step7: Total waiting time calculated by adding the waiting time of each process.**

**Step8: Total turnaround time calculated by adding all total time of each process.**

**Step9: Calculate average waiting time by dividing the total waiting time by total number of process.**

**Step10: Calculate average turnaround time by dividing the total waiting time by total number of process.**

**Step11: Display the result**

**Code:-**

#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("\nEnter Burst Time:\n");

for(i=0;i<n;i++)

{

printf("P[%d]:",i+1);

scanf("%d",&bt[i]);

p[i]=i+1;

}

//sorting burst time in ascending order using selection sort

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; //waiting time zero initially

//calculate waiting time

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; //average waiting time

total=0;

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

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i]; //calculate turnaround time

total+=tat[i];

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

}

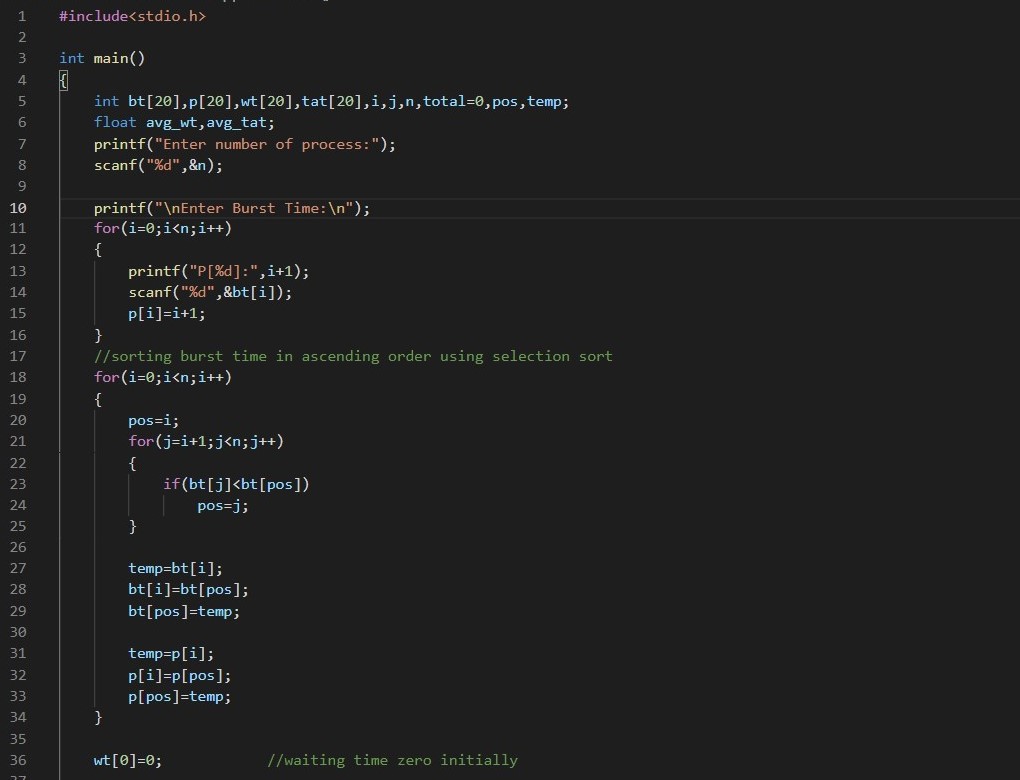
avg\_tat=(float)total/n; //average turnaround time

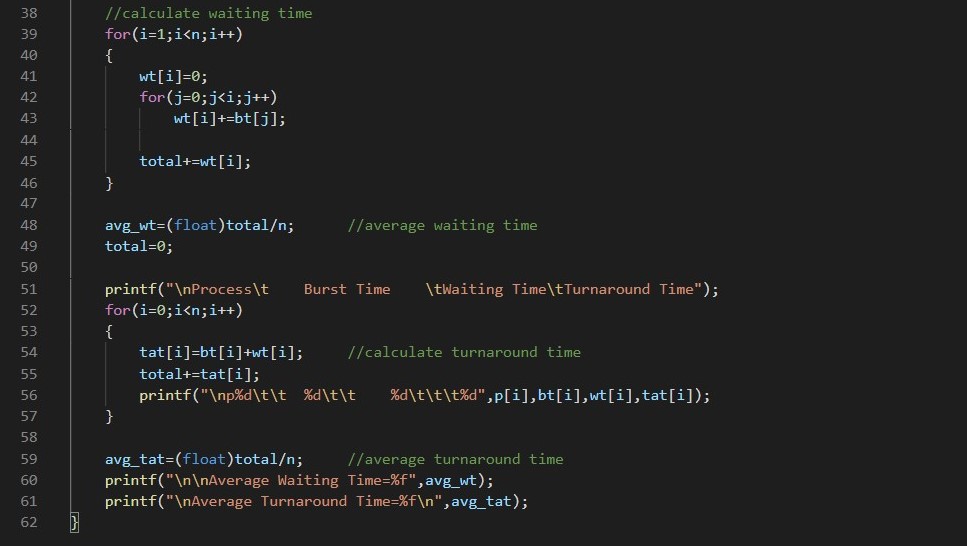
printf("\n\nAverage Waiting Time=%f",avg\_wt);

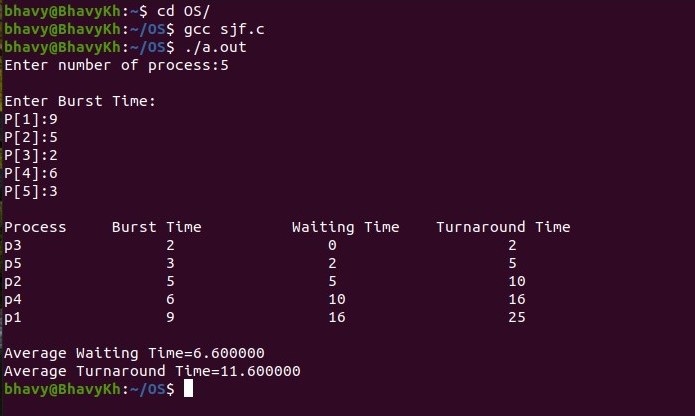
printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

**Screenshot/Output:-**

****

****

****

**\*\*\*\*\*\*\*End of Exercise - 2.2\*\*\*\*\*\*\***

OPERATING SYSTEM LAB

EXERCISE – 2.3

ROUND ROBIN

**AIM: To write a C program to simulate the CPU scheduling algorithm for round robin PROBLEM.**

**THEORY:**

**CPU scheduler will decide which process should be given the CPU for its execution .For this it use different algorithm to choose among the process .one among that algorithm is Round robin algorithm. In this algorithm we are assigning some time slice .The process is allocated according to the time slice ,if the process service time is less than the time slice then process itself will release the CPU voluntarily .The scheduler will then proceed to the next process in the ready queue .If the CPU burst of the currently running process is longer than time quantum ,the timer will go off and will cause an interrupt to the operating system .A context switch will be executed and the process will be put at the tail of the ready queue.**

**Algorithm:-**

**Step 1: Initialize all the structure elements.**

**Step 2: Receive inputs from the user to fill process id, burst time and arrival time.**

**Step 3: Calculate the waiting time for all the process id.**

1. **The waiting time for first instance of a process is calculated as: a[i].waittime=count + a[i].arrivt**
2. **ii) The waiting time for the rest of the instances of the process is calculated as:**

**a) If the time quantum is greater than the remaining burst time then waiting time is calculated as: a[i].waittime=count + tq**

**b) Else if the time quantum is greater than the remaining burst time then waiting time is calculated as: a[i].waittime=count - remaining burst time**

**Step 4: Calculate the average waiting time and average turnaround time**

**Step 5: Display the result.**

**Code:-**

#include<stdio.h>

int main()

{

int i, limit, total = 0, x, count = 0, tq;

int wt=0, tt=0, at[20], bt[20], temp[20];

float awt, att;

printf("\nEnter Total Number of Processes:\t");

scanf("%d", &limit);

x = limit;

for(i = 0; i < limit; i++)

{

printf("\nEnter Details of Process[%d]\n", i + 1);

printf("Arrival Time:\t");

scanf("%d", &at[i]);

printf("Burst Time:\t");

scanf("%d", &bt[i]);

temp[i] = bt[i];

}

printf("\nEnter Time Quantum:\t");

scanf("%d", &tq);

printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

for(total = 0, i = 0; x != 0;)

{

if(temp[i] <= tq && temp[i] > 0)

{

total = total + temp[i];

temp[i] = 0;

count = 1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - tq;

total = total + tq;

}

if(temp[i] == 0 && count == 1)

{

x--;

printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, bt[i], total - at[i], total - at[i] - bt[i]);

wt = wt + total - at[i] - bt[i];

tt = tt + total - at[i];

count = 0;

}

if(i == limit - 1)

{

i = 0;

}

else if(at[i + 1] <= total)

{

i++;

}

else

{

i = 0;

}

}

awt = wt\* 1.0 / limit;

att = tt \* 1.0 / limit;

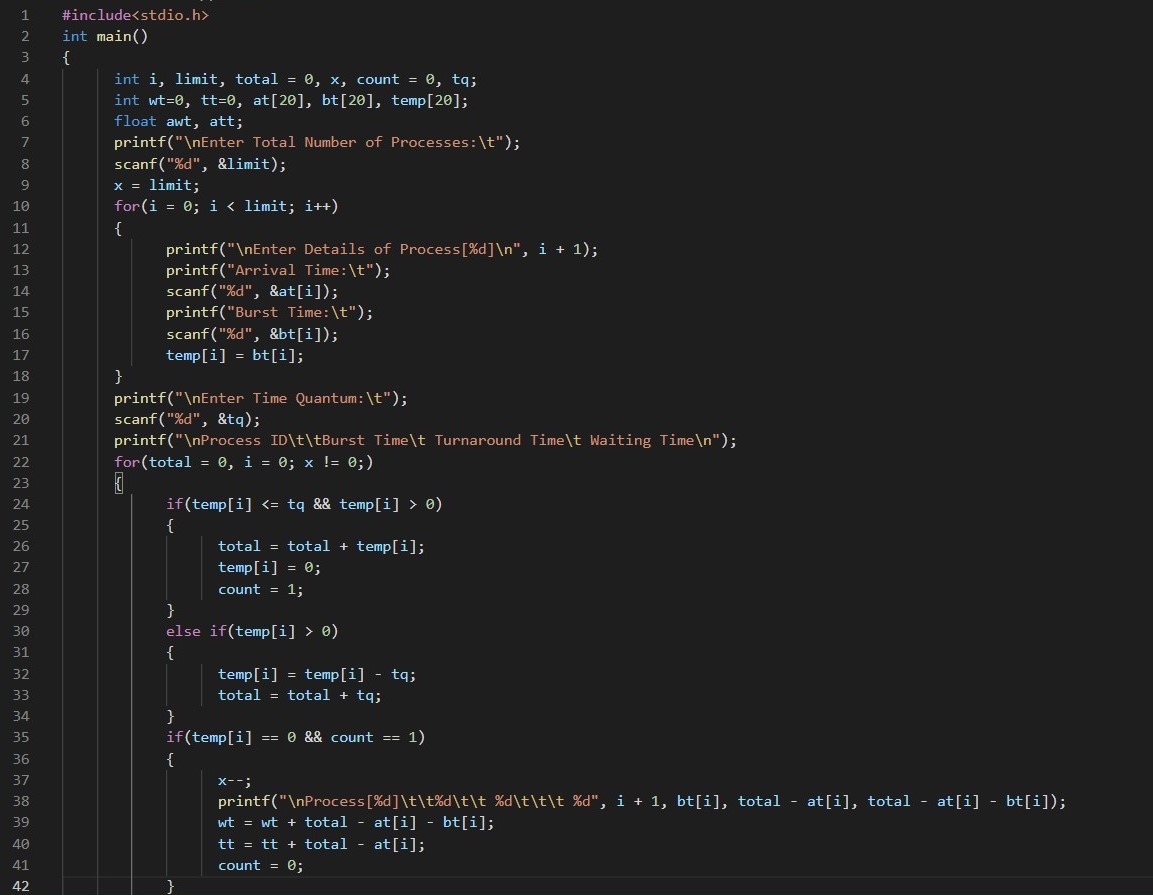
printf("\n\nAverage Waiting Time:\t%f", awt);

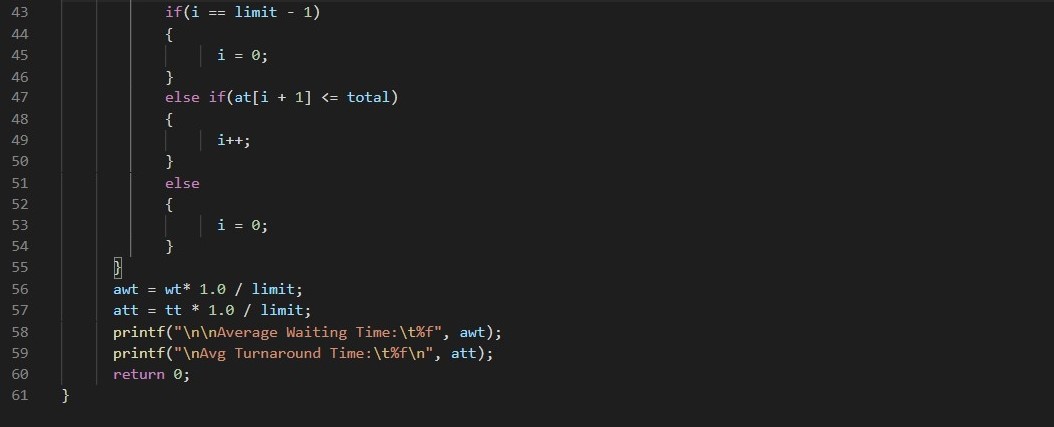
printf("\nAvg Turnaround Time:\t%f\n", att);

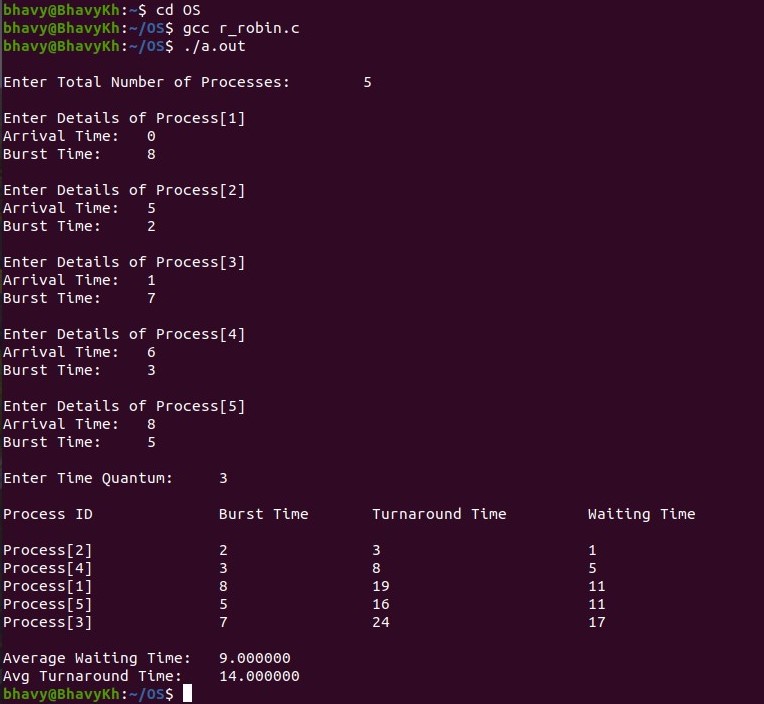
return 0;

}

**Screenshot/Output:-**

****

****

****

**\*\*\*\*\*\*\*End of Exercise - 2.3\*\*\*\*\*\*\***

**THANK YOU**