**EXERCISE – 4**

**Name – Bhavy Kharbanda**

**SAP ID – 500082531**

**Roll Number – R214220342**

**Batch – B11**

**Date of Performance – 12/04/2021**

**Date of Submission – 23/04/2021**

OPERATING SYSTEM LAB

EXERCISE – 4.1

ROUND ROBIN

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

**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;

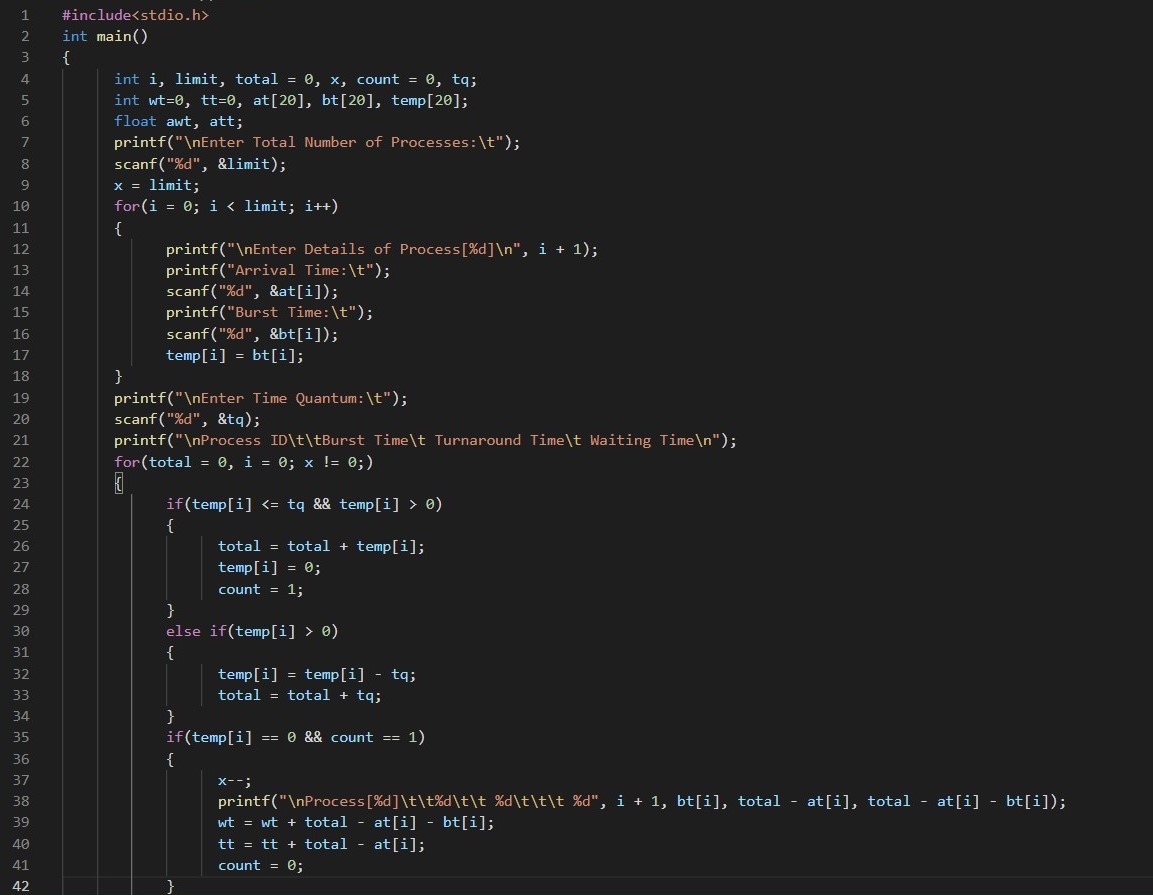
printf("\n\nAverage Waiting Time:\t%f", awt);

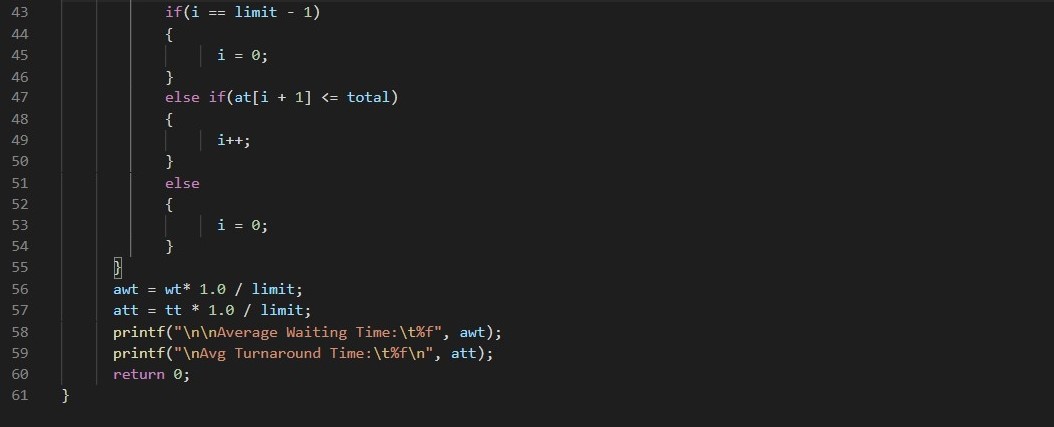
printf("\nAvg Turnaround Time:\t%f\n", att);

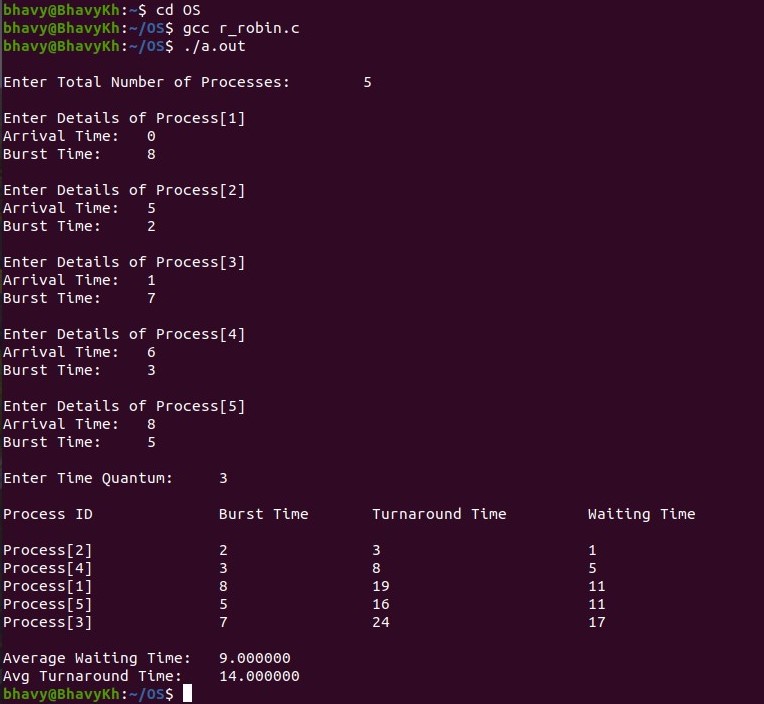
return 0;

}

**Screenshot/Output:-**

****

****

****

**\*\*\*\*\*\*\*End of Exercise – 4.1\*\*\*\*\*\*\***

OPERATING SYSTEM LAB

EXERCISE – 4.2

PRIORITY SCHEDULING

**AIM: To write a C program to simulate the CPU scheduling algorithm for priority scheduling with arrival time.**

**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.**

**Code:-**

#include <stdio.h>

#include <conio.h>

int main() {

int et[20], at[10], n, i, j, temp, p[10], st[10], ft[10], wt[10], ta[10];

int totwt = 0, totta = 0;

float awt, ata;

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

scanf("%d", &n);

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

printf("Enter Arrival Time, Burst Time & priority:\n");

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

}

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

for (j = 0; j < n; j++) {

if (p[i] < p[j]) {

temp = p[i];

p[i] = p[j];

p[j] = temp;

temp = at[i];

at[i] = at[j];

at[j] = temp;

temp = et[i];

et[i] = et[j];

et[j] = temp;

}

}

}

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

if (i == 0) {

st[i] = at[i];

wt[i] = st[i] - at[i];

ft[i] = st[i] + et[i];

ta[i] = ft[i] - at[i];

} else {

st[i] = ft[i - 1];

wt[i] = st[i] - at[i];

ft[i] = st[i] + et[i];

ta[i] = ft[i] - at[i];

}

totwt += wt[i];

totta += ta[i];

}

awt = (float)totwt / n;

ata = (float)totta / n;

printf("\nArrival Time\tBurst Time\tPriority\tWaiting Time\tTA Time");

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

printf("\n%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d", at[i], et[i], p[i], wt[i], ta[i]);

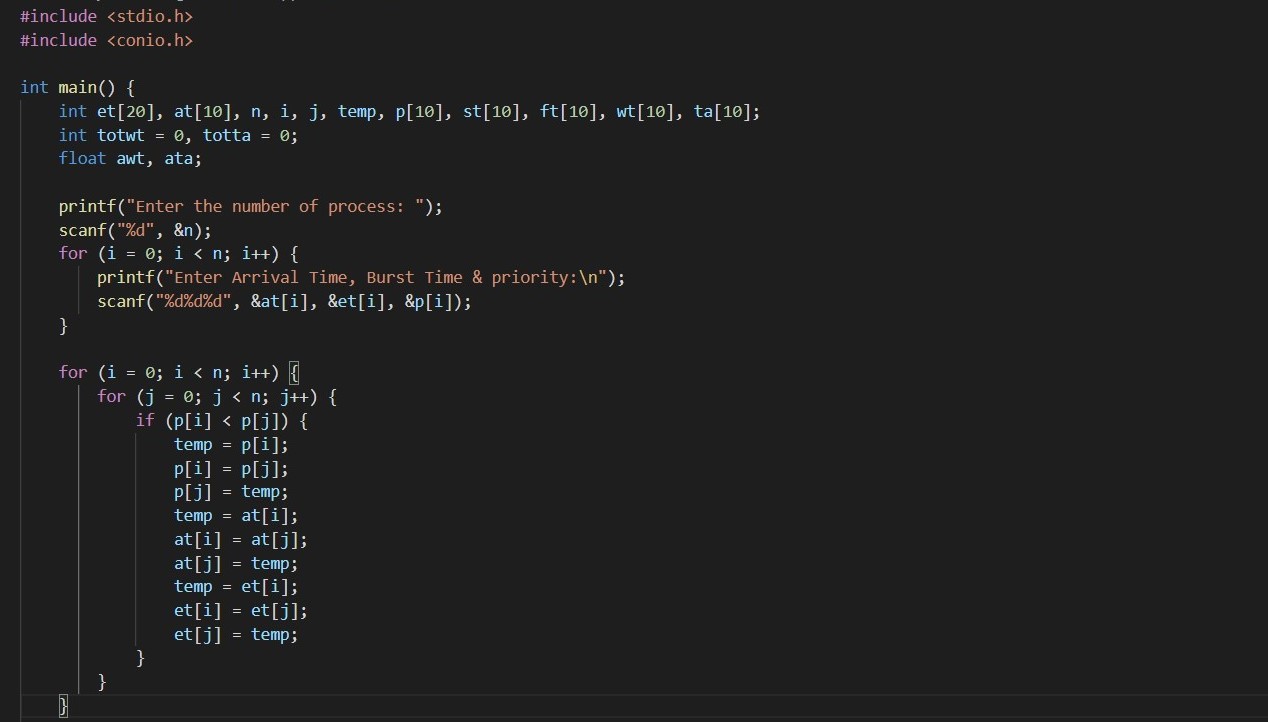
printf("\nAverage Waiting Time : %.2f", awt);

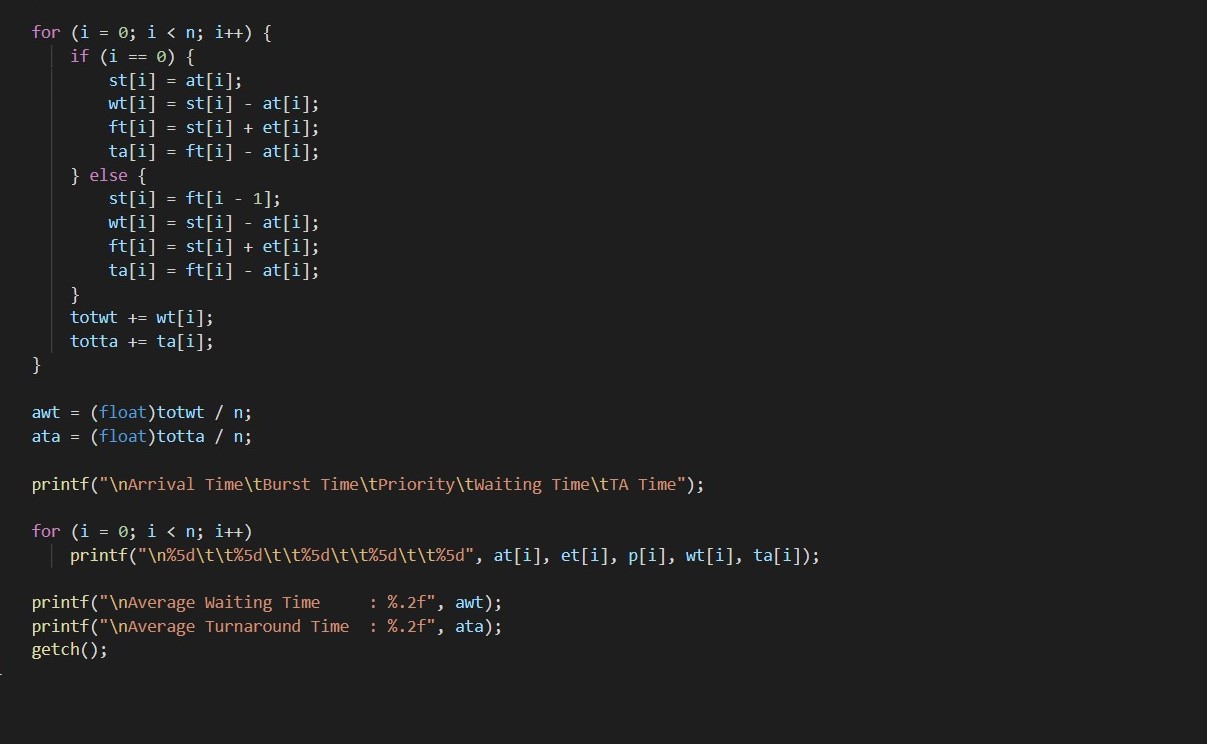
printf("\nAverage Turnaround Time : %.2f", ata);

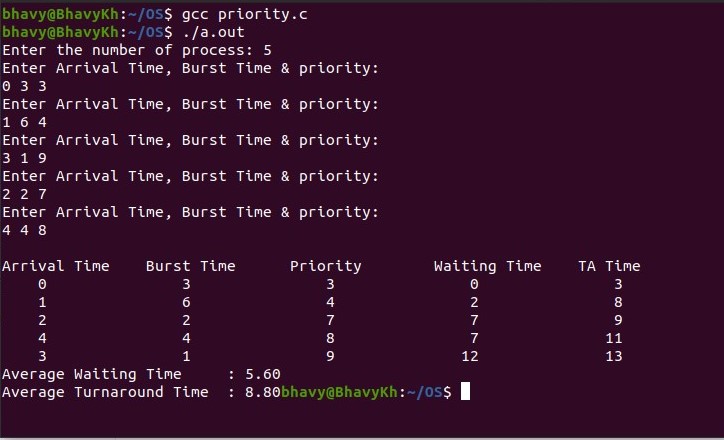
getch();

}

**Screenshot/Output:-**





****

**\*\*\*\*\*\*\*End of Exercise – 4.2\*\*\*\*\*\*\***

**THANK YOU**