Name: Aditya Matale

Roll. No: 25

PRN: 12110037

Class: AIDS – TY (B) Batch_01

OS LAB

Scheduling Algorithms

#] **Menu Driven Program:**

1] **First-Come, First-Served (FCFS)**

2] **Shortest Job First (SJF)**

3] **Priority Scheduling**

4] **Round Robin (RR)**

CODE:

```
#include<stdio.h>

void FCFS()
{
 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",att);
}

void SJF()
{

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",avg_tat);
}

void RR()
{
//Input no of processed
int n;
printf("Enter Total Number of Processes:");
scanf("%d", &n);

int wait_time = 0, ta_time = 0, arr_time[n], burst_time[n], temp_burst_time[n];
int x = n;

//Input details of processes
for(int i = 0; i < n; i++)
```

```
{
printf("Enter Details of Process %d \n", i + 1);
printf("Arrival Time: ");
scanf("%d", &arr_time[i]);
printf("Burst Time: ");
scanf("%d", &burst_time[i]);
temp_burst_time[i] = burst_time[i];
}

//Input time slot
int time_slot;
printf("Enter Time Slot:");
scanf("%d", &time_slot);

//Total indicates total time
//counter indicates which process is executed
int total = 0, counter = 0,i;

printf("Process ID Burst Time Turnaround Time Waiting Time\n");
for(total=0, i = 0; x!=0; )
{
// define the conditions
if(temp_burst_time[i] <= time_slot && temp_burst_time[i] > 0)
{
total = total + temp_burst_time[i];
temp_burst_time[i] = 0;
counter=1;
}
else if(temp_burst_time[i] > 0)
{
temp_burst_time[i] = temp_burst_time[i] - time_slot;
total += time_slot;
```

```
}
if(temp_burst_time[i]==0 && counter==1)
{
x--; //decrement the process no.
printf("\nProcess No %d \t\t %d\t\t\t %d\t\t\t %d\t\t", i+1, burst_time[i],
total-arr_time[i], total-arr_time[i]-burst_time[i]);
wait_time = wait_time+total-arr_time[i]-burst_time[i];
ta_time += total -arr_time[i];
counter =0;
}
if(i==n-1)
{
i=0;
}
else if(arr_time[i+1]<=total)
{
i++;
}
else
{
i=0;
}
}

float average_wait_time = wait_time * 1.0 / n;
float average_turnaround_time = ta_time * 1.0 / n;

printf("\nAverage Waiting Time:%f", average_wait_time);
printf("\nAvg Turnaround Time:%f", average_turnaround_time);

//  return 0;
}
```

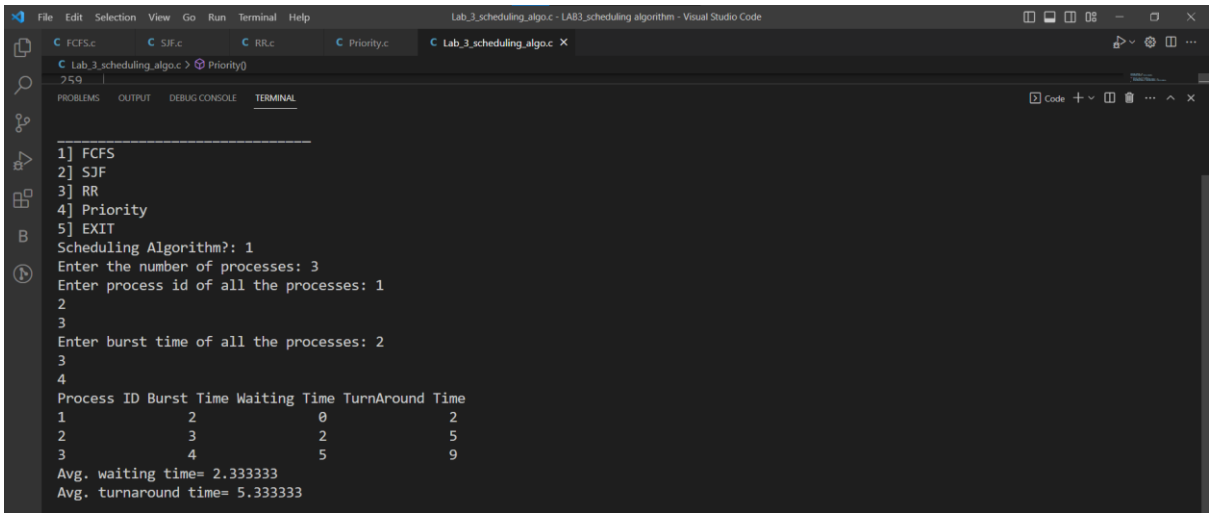
```
void Priority(){
   int n;
   printf("Enter the number of processes: ");
   scanf("%d", &n);

   int processes[n];
   int burst_time[n];
   int priority[n];

   // Input process details
   for (int i = 0; i < n; i++) {
      printf("Enter the details for Process %d:\n", i + 1);
      processes[i] = i + 1;
      printf("Burst Time: ");
      scanf("%d", &burst_time[i]);
      printf("Priority: ");
      scanf("%d", &priority[i]);
   }

   // Sort the processes based on priority (higher priority first) using Bubble Sort
   for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
         if (priority[j] > priority[j + 1]) {
            // Swap the processes
            int temp = processes[j];
            processes[j] = processes[j + 1];
            processes[j + 1] = temp;

            temp = burst_time[j];
            burst_time[j] = burst_time[j + 1];
            burst_time[j + 1] = temp;
```

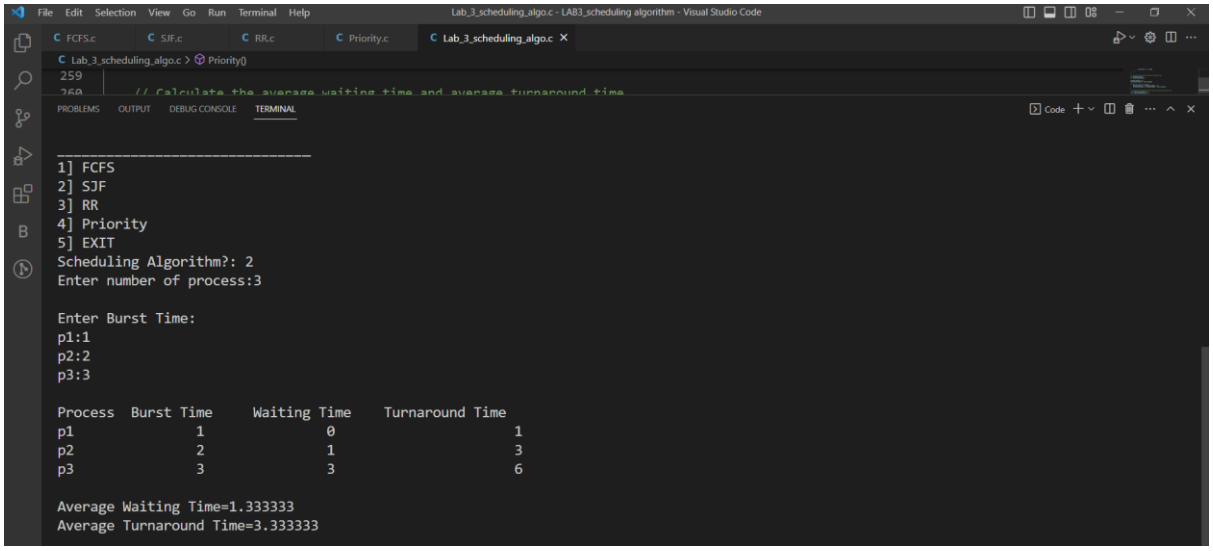
```
            temp = priority[j];
            priority[j] = priority[j + 1];
            priority[j + 1] = temp;
        }
    }
}

// Calculate the waiting time and turnaround time
int waiting_time[n];
int turnaround_time[n];

waiting_time[0] = 0;
turnaround_time[0] = burst_time[0];

for (int i = 1; i < n; i++) {
    waiting_time[i] = turnaround_time[i - 1];
    turnaround_time[i] = waiting_time[i] + burst_time[i];
}

// Calculate the average waiting time and average turnaround time
float avg_waiting_time = 0;
float avg_turnaround_time = 0;

for (int i = 0; i < n; i++) {
    avg_waiting_time += waiting_time[i];
    avg_turnaround_time += turnaround_time[i];
}

avg_waiting_time /= n;
avg_turnaround_time /= n;
```

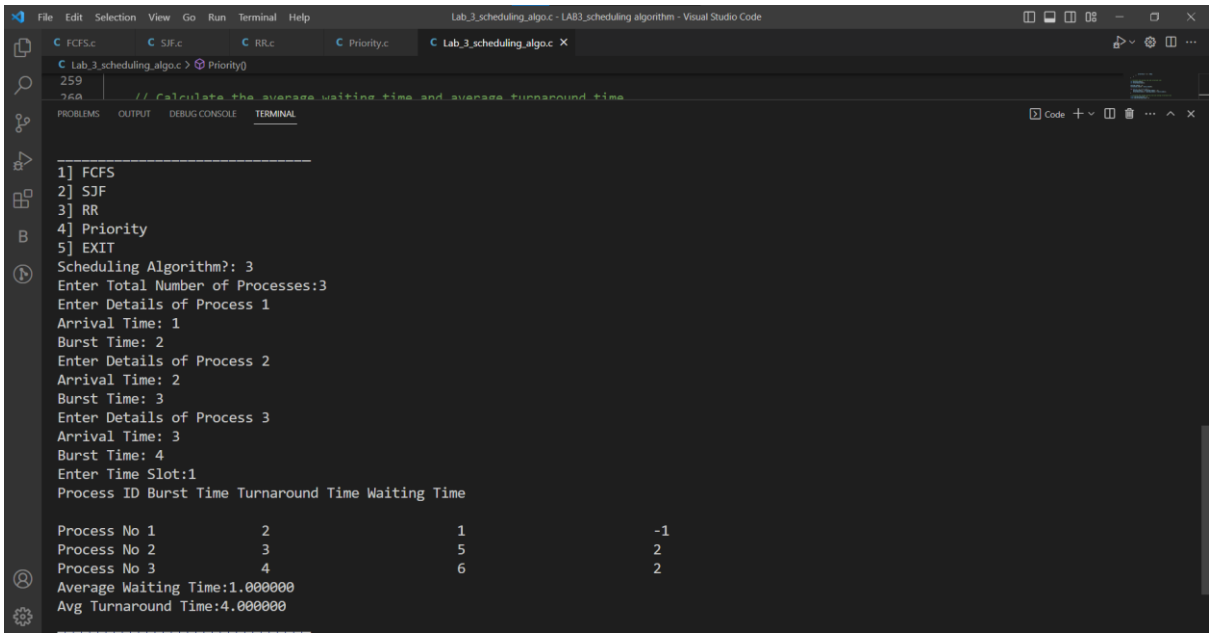
```
    // Display the results
    printf("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], burst_time[i], priority[i],
waiting_time[i], turnaround_time[i]);
    }

    printf("Average Waiting Time: %.2f\n", avg_waiting_time);
    printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);

}

int main(){
    while (1)
    {

    printf("\n_____\n1] FCFS\n2] SJF\n3] RR\n4] Priority\n5]
EXIT\nScheduling Algorithm?: ");
    int choice;
    scanf("%d",&choice);
    if (choice==5){
        break;
    }
    switch (choice)
    {
    case 1:
        FCFS();
        break;
    case 2:
        SJF();
        break;
    case 3:
```

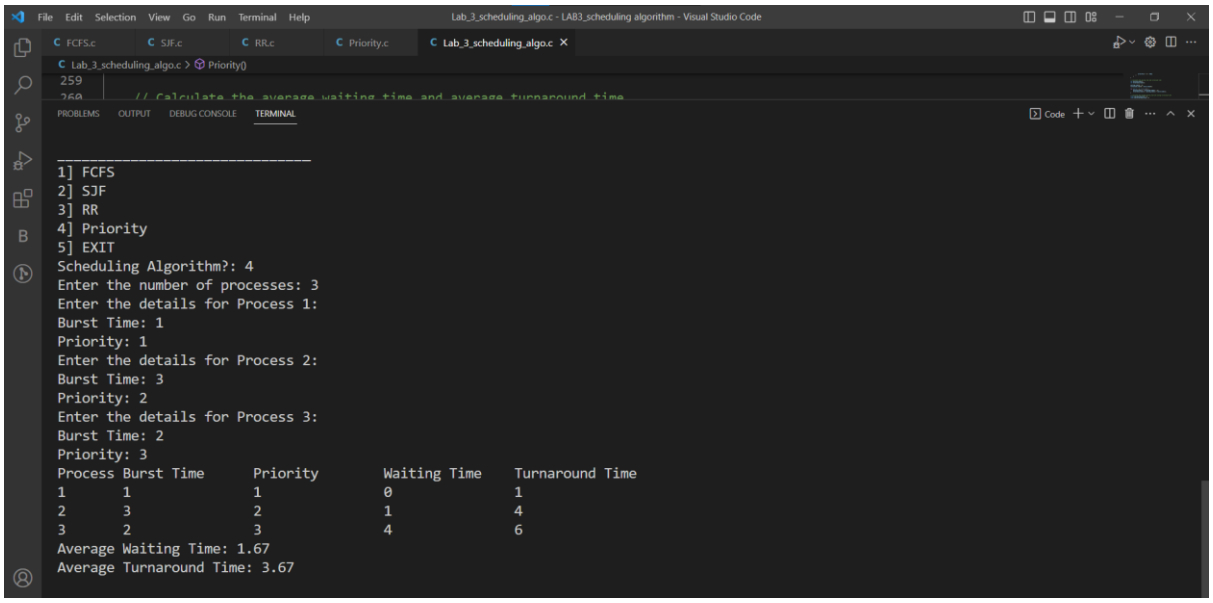
```
        RR();
        break;
    case 4:
        Priority();
        break;

    default:
        printf("invalid choice");
        break;
    }
    }

    return 0;
}
```

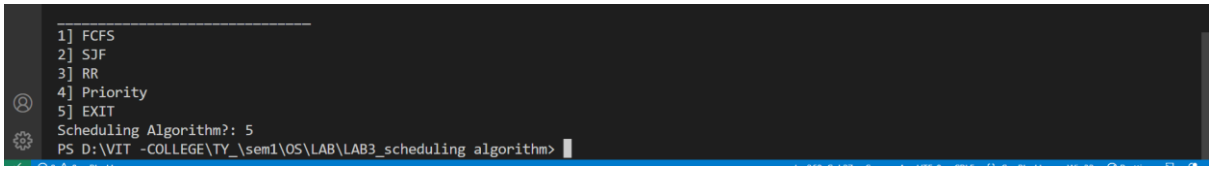
OUTPUT:




Lab_3_scheduling_algo.c - LAB3_scheduling algorithm - Visual Studio Code
FCFS.c
SJF.c
RR.c
Priority.c
Lab_3_scheduling_algo.c
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1] FCFS
2] SJF
3] RR
4] Priority
5] EXIT
Scheduling Algorithm?: 2
Enter number of process:3
Enter Burst Time:
p1:1
p2:2
p3:3
Process Burst Time Waiting Time Turnaround Time
p1 1 0 1
p2 2 1 3
p3 3 3 6
Average Waiting Time=1.333333
Average Turnaround Time=3.333333


Lab_3_scheduling_algo.c - LAB3_scheduling algorithm - Visual Studio Code
FCFS.c
SJF.c
RR.c
Priority.c
Lab_3_scheduling_algo.c
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1] FCFS
2] SJF
3] RR
4] Priority
5] EXIT
Scheduling Algorithm?: 3
Enter Total Number of Processes:3
Enter Details of Process 1
Arrival Time: 1
Burst Time: 2
Enter Details of Process 2
Arrival Time: 2
Burst Time: 3
Enter Details of Process 3
Arrival Time: 3
Burst Time: 4
Enter Time Slot:1
Process ID Burst Time Turnaround Time Waiting Time
Process No 1 2 1 -1
Process No 2 3 5 2
Process No 3 4 6 2
Average Waiting Time:1.000000
Avg Turnaround Time:4.000000


Lab_3_scheduling_algo.c - LAB3_scheduling algorithm - Visual Studio Code
FCFS.c
SJF.c
RR.c
Priority.c
Lab_3_scheduling_algo.c
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1] FCFS
2] SJF
3] RR
4] Priority
5] EXIT
Scheduling Algorithm?: 4
Enter the number of processes: 3
Enter the details for Process 1:
Burst Time: 1
Priority: 1
Enter the details for Process 2:
Burst Time: 3
Priority: 2
Enter the details for Process 3:
Burst Time: 2
Priority: 3
Process Burst Time Priority Waiting Time Turnaround Time
1 1 1 0 1
2 3 2 1 4
3 2 3 4 6
Average Waiting Time: 1.67
Average Turnaround Time: 3.67


1] FCFS
2] SJF
3] RR
4] Priority
5] EXIT
Scheduling Algorithm?: 5
PS D:\VIT -COLLEGE\TY_\sem1\OS\LAB\LAB3_scheduling algorithm>