

CSA0496-OPERATING SYSTEM WITH TASK MIGRATION

DAY-01

NAME: P.PRANAVI

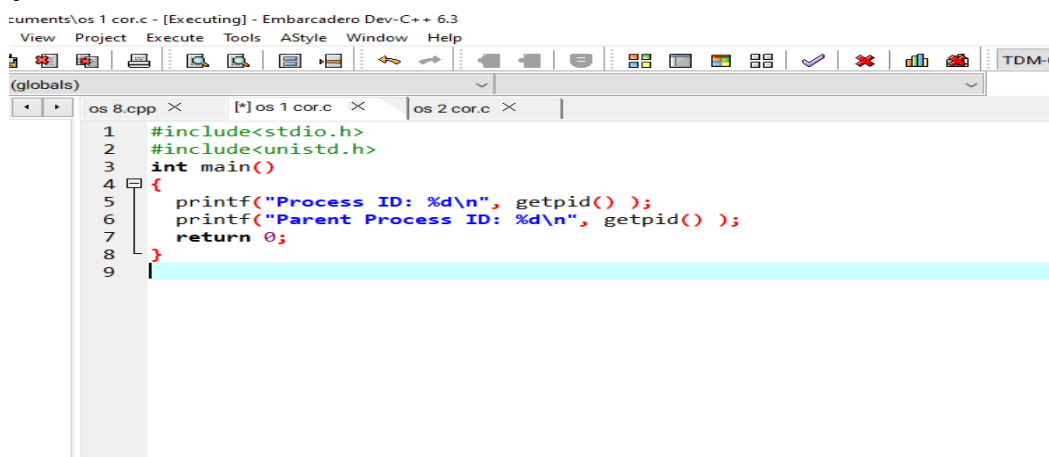
REG.NO:192110530

1.Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

Program:

```
#include<stdio.h>
#include<unistd.h>
int main()
{
    printf("Process ID: %d\n", getpid() );
    printf("Parent Process ID: %d\n", getppid() );
    return 0;
}
```

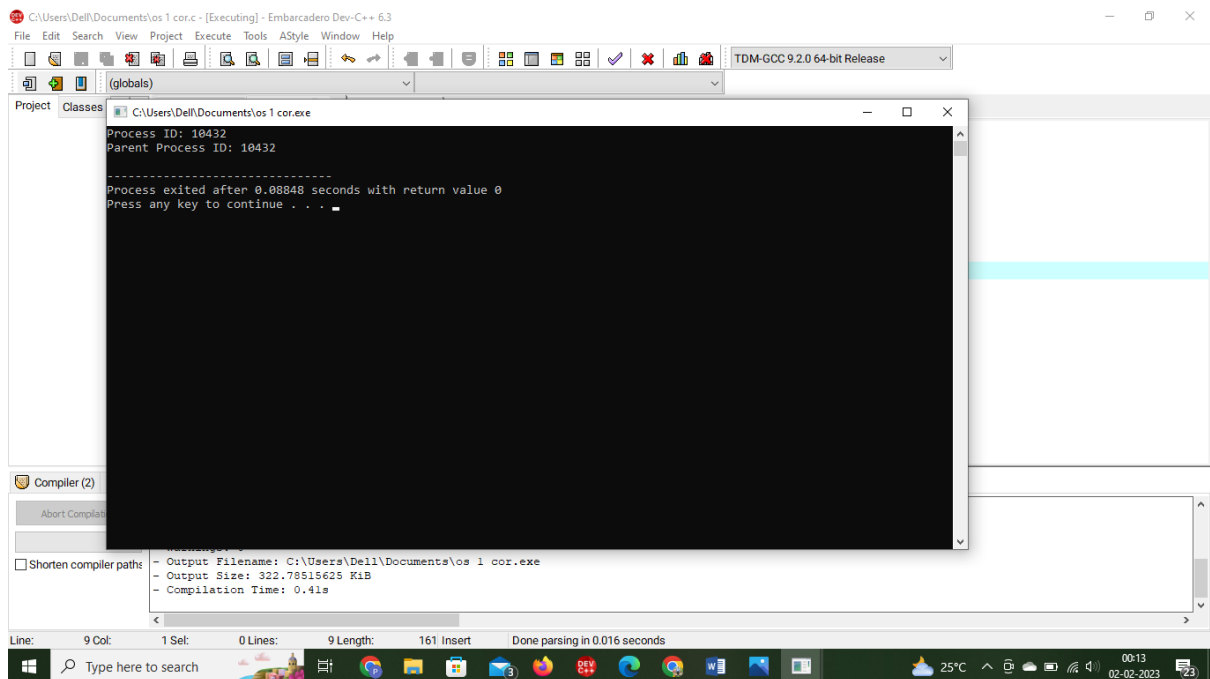
Input:

A screenshot of an IDE window titled "documents\os 1 cor.c - [Executing] - Embarcadero Dev-C++ 6.3". The window shows a C program with the following code:

```
1 #include<stdio.h>
2 #include<unistd.h>
3 int main()
4 {
5     printf("Process ID: %d\n", getpid() );
6     printf("Parent Process ID: %d\n", getppid() );
7     return 0;
8 }
9
```

The code is displayed in a window titled "os 1 cor.c". The IDE interface includes a menu bar (View, Project, Execute, Tools, AStyle, Window, Help) and a toolbar with various icons for file operations and execution. The code is written in a monospaced font with syntax highlighting.

Output:



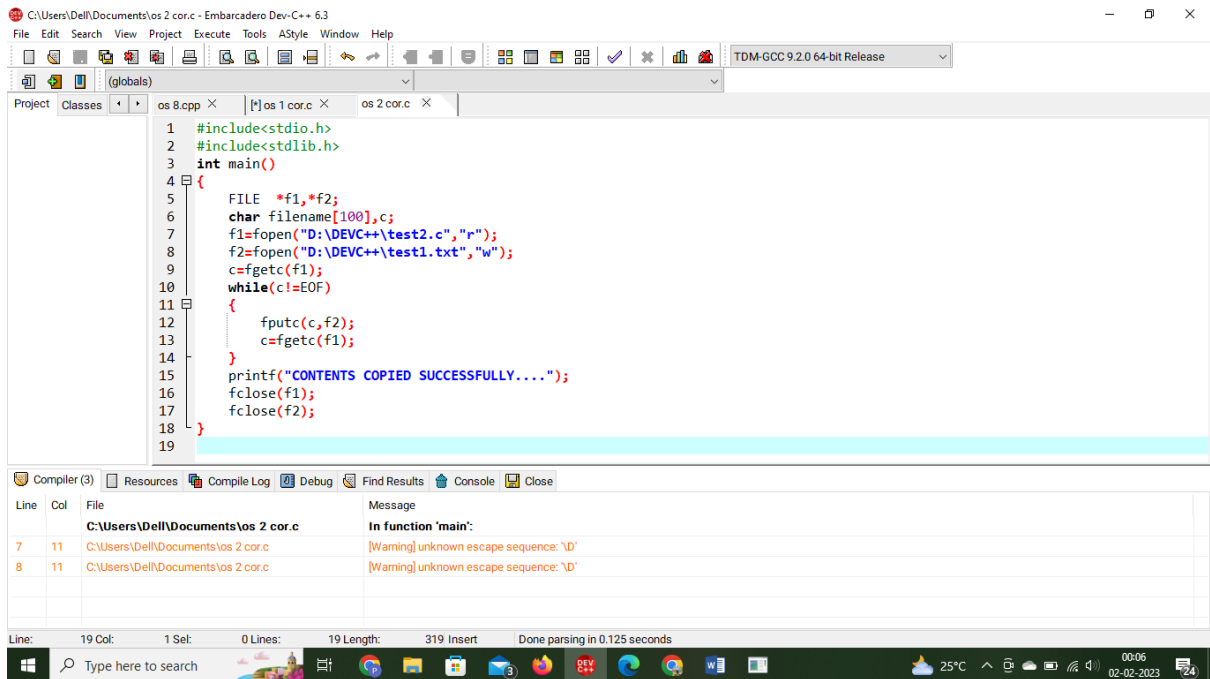
2. Identify the system calls to copy the content of one file to another and illustrate the same using a C program.

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    FILE *f1,*f2;
    char filename[100],c;
    f1=fopen("D:\\DEVCC++\\test2.c","r");
    f2=fopen("D:\\DEVCC++\\test1.txt","w");
    c=fgetc(f1);
    while(c!=EOF)
    {
        fputc(c,f2);
        c=fgetc(f1);
    }
    printf("CONTENTS COPIED SUCCESSFULLY....");
    fclose(f1);
    fclose(f2);
}
```

}

Input:

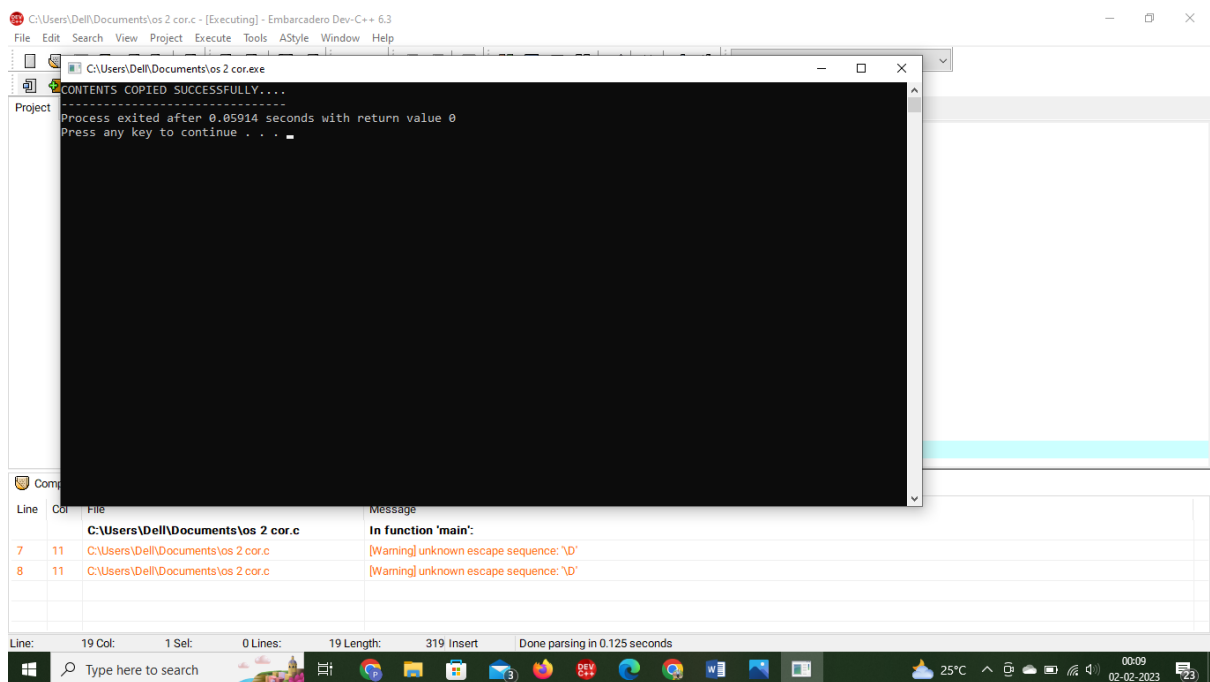


The screenshot shows the Embarcadero Dev-C++ 6.3 IDE. The main window displays the source code for 'os 2 cor.c'. The code includes `<stdio.h>` and `<stdlib.h>`, and defines a `main` function. Inside `main`, it opens two files: `D:\DEV++\test2.c` for reading and `D:\DEV++\test1.txt` for writing. It then copies the contents of `test2.c` to `test1.txt` character by character until the end of the file is reached. The program prints "CONTENTS COPIED SUCCESSFULLY...." and closes both files.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main()
4 {
5     FILE *f1,*f2;
6     char filename[100],c;
7     f1=fopen("D:\DEV++\test2.c","r");
8     f2=fopen("D:\DEV++\test1.txt","w");
9     c=fgetc(f1);
10    while(c!=EOF)
11    {
12        fputc(c,f2);
13        c=fgetc(f1);
14    }
15    printf("CONTENTS COPIED SUCCESSFULLY....");
16    fclose(f1);
17    fclose(f2);
18 }
19
```

The Compiler window shows two warnings in the `main` function: "[Warning] unknown escape sequence: '\D'" on lines 7 and 8.

Output:



The screenshot shows the same IDE with the execution output window open. The output displays the message "CONTENTS COPIED SUCCESSFULLY...." followed by a series of dots. Below this, it states "Process exited after 0.05914 seconds with return value 0" and "Press any key to continue . . .". The Compiler window still shows the same two warnings as in the previous screenshot.

```
CONTENTS COPIED SUCCESSFULLY....
.....
Process exited after 0.05914 seconds with return value 0
Press any key to continue . . .
```

3.Design a CPU scheduling program with C using First Come First Served technique with the following considerations.

a. All processes are activated at time 0.

b. Assume that no process waits on I/O devices.

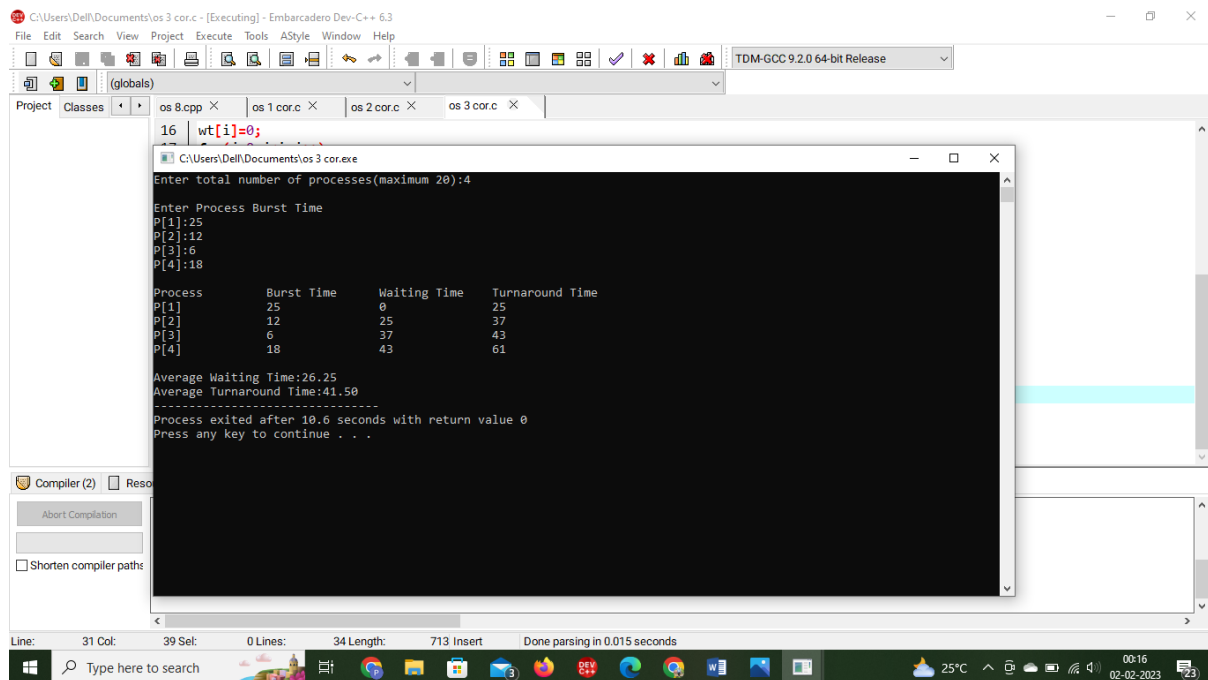
Program:

```
#include<stdio.h>

int main()
{
    int n,bt[20],wt[20],tat[20],i,j;
    float avwt=0,avtat=0;
    printf("Enter total number of processes(maximum 20):");
    scanf("%d",&n);
    printf("\nEnter Process Burst Time\n");
    for(i=0;i<n;i++)
    {
        printf("P[%d]:",i+1);
        scanf("%d",&bt[i]);
    }
    wt[0]=0;for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }
    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
```

Input:

Output:



```
16 wt[i]=0;
C:\Users\Devl\Documents\os 3 cor.exe
Enter total number of processes(maximum 20):4
Enter Process Burst Time
P[1]:25
P[2]:12
P[3]:6
P[4]:18
Process      Burst Time  Waiting Time  Turnaround Time
P[1]         25          0             25
P[2]         12          25            37
P[3]          6          37            43
P[4]         18          43            61
Average Waiting Time:26.25
Average Turnaround Time:41.50
-----
Process exited after 10.6 seconds with return value 0
Press any key to continue . . .
```

4. Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

Program:

```
#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;
}
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("\nProcess\tBurst Time \tWaiting Time \tTurnaround Time\n");

for(i=0;i<n;i++)
{
    tat[i]=bt[i]+wt[i];

    total+=tat[i];

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

avg_tat=(float)total/n;

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

printf("\nAverage Turnaround Time=%f\n",avg_tat);
}

```

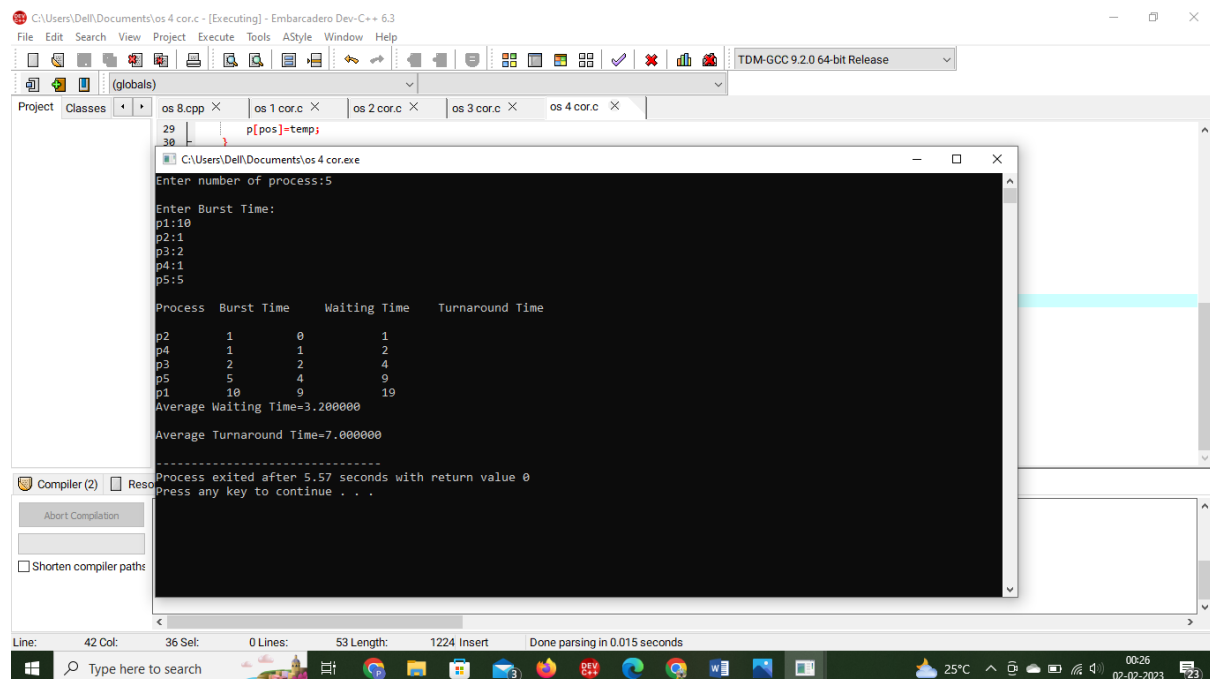
Input:

```

1 #include<iostream.h>
2 int main()
3 {
4     int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
5     float avg_wt,avg_tat;
6     printf("Enter number of process:\n");
7     scanf("%d",&n);
8     printf("Enter Burst Time:\n");
9     for(i=0;i<n;i++)
10     {
11         printf("p%d:",i+1);
12         scanf("%d",&bt[i]);
13         p[i]=i+1;
14     }
15     for(i=0;i<n;i++)
16     {
17         pos=i;
18         for(j=i+1;j<n;j++)
19         {
20             if(bt[j]<bt[pos])
21             {
22                 pos=j;
23             }
24             temp=bt[i];
25             bt[i]=bt[pos];
26             bt[pos]=temp;
27             temp=p[i];
28             p[i]=p[pos];
29             p[pos]=temp;
30         }
31         wt[i]=0;
32         for(i=1;i<n;i++)
33         {
34             wt[i]=0;
35             for(j=0;j<i;j++)
36             {
37                 wt[i]=bt[j];
38             }
39             total+=wt[i];
40         }
41         avg_wt=(float)total/n;
42         total=0;
43         printf("\nProcess\tBurst Time \tWaiting Time \tTurnaround Time\n");
44         for(i=0;i<n;i++)
45         {
46             tat[i]=bt[i]+wt[i];
47             total+=tat[i];
48             printf("\np%d\t\t %d\t\t %d\t\t %d",p[i],bt[i],wt[i],tat[i]);
49         }
50         avg_tat=(float)total/n;
51         printf("\nAverage Waiting Time=%f\n",avg_wt);
52         printf("\nAverage Turnaround Time=%f\n",avg_tat);
53     }
54 }

```


Output:



```
C:\Users\De\Documents\os 4 cor.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes os 8.cpp x os 1 cor.c x os 2 cor.c x os 3 cor.c x os 4 cor.c x
29 p[pos]=temp;
30
C:\Users\De\Documents\os 4 cor.exe
Enter number of process:5
Enter Burst Time:
p1:10
p2:1
p3:2
p4:1
p5:5
Process Burst Time Waiting Time Turnaround Time
p2 1 0 1
p4 1 1 2
p3 2 2 4
p5 5 4 9
p1 10 9 19
Average Waiting Time=3.200000
Average Turnaround Time=7.000000
-----
Process exited after 5.57 seconds with return value 0
Press any key to continue . . .
```

5. Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.

Program:

```
#include<stdio.h>
```

```
struct priority_scheduling {
```

```
    char process_name;
```

```
    int burst_time;
```

```
    int waiting_time;
```

```
    int turn_around_time;
```

```
    int priority;
```

```
};
```

```
int main() {
```

```
    int number_of_process;
```

```
    int total = 0;
```

```
    struct priority_scheduling temp_process;
```

```
    int ASCII_number = 65;
```

```

int position;

float average_waiting_time;

float average_turnaround_time;

printf("Enter the total number of Processes: ");

scanf("%d", & number_of_process);

struct priority_scheduling process[number_of_process];

printf("\nPlease Enter the Burst Time and Priority of each process:\n");

for (int i = 0; i < number_of_process; i++) {

    process[i].process_name = (char) ASCII_number;

    printf("\nEnter the details of the process %c \n", process[i].process_name);

    printf("Enter the burst time: ");

    scanf("%d", & process[i].burst_time);

    printf("Enter the priority: ");

    scanf("%d", & process[i].priority);

    ASCII_number++;

}

for (int i = 0; i < number_of_process; i++) {

    position = i;

    for (int j = i + 1; j < number_of_process; j++) {

        if (process[j].priority > process[position].priority)

            position = j;

    }

    temp_process = process[i];

    process[i] = process[position];

    process[position] = temp_process;

}

process[0].waiting_time = 0;

for (int i = 1; i < number_of_process; i++) {

    process[i].waiting_time = 0;

```

```

    for (int j = 0; j < i; j++) {
        process[i].waiting_time += process[j].burst_time;
    }
    total += process[i].waiting_time;
}

average_waiting_time = (float) total / (float) number_of_process;
total = 0;

printf("\n\nProcess_name \t Burst Time \t Waiting Time \t Turnaround Time\n");
printf("-----\n");

for (int i = 0; i < number_of_process; i++) {
    process[i].turn_around_time = process[i].burst_time + process[i].waiting_time;
    total += process[i].turn_around_time;

    printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process_name, process[i].burst_time,
process[i].waiting_time, process[i].turn_around_time);

    printf("\n-----\n");
}

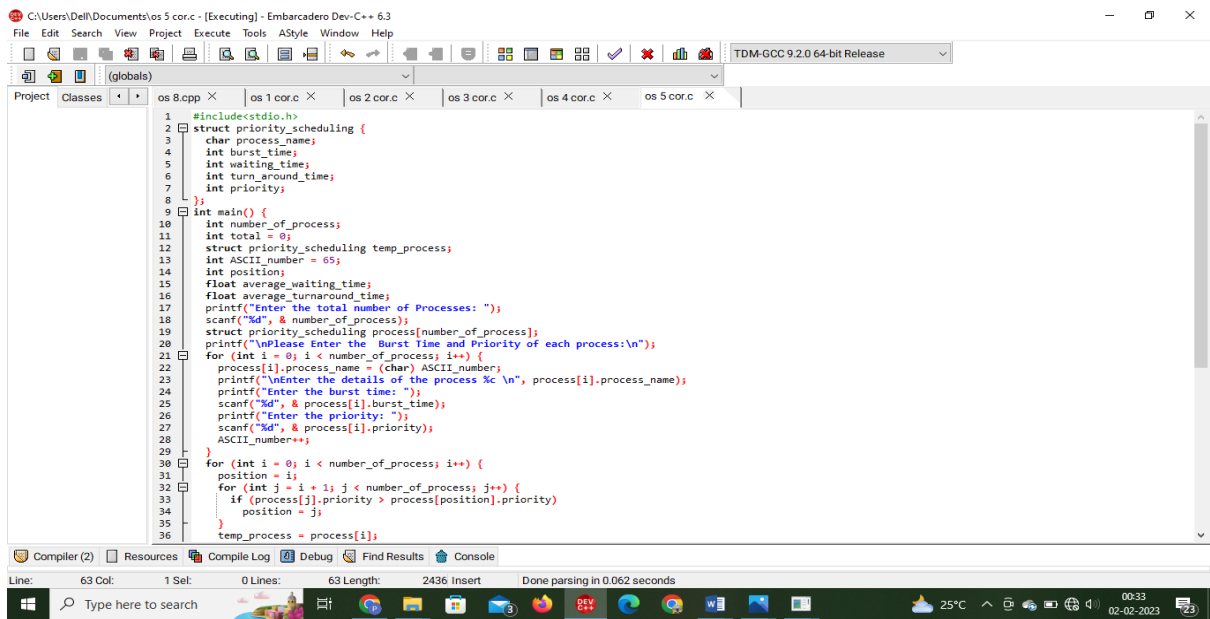
average_turnaround_time = (float) total / (float) number_of_process;
printf("\n\n Average Waiting Time : %f", average_waiting_time);
printf("\n\n Average Turnaround Time: %f\n", average_turnaround_time);

return 0;

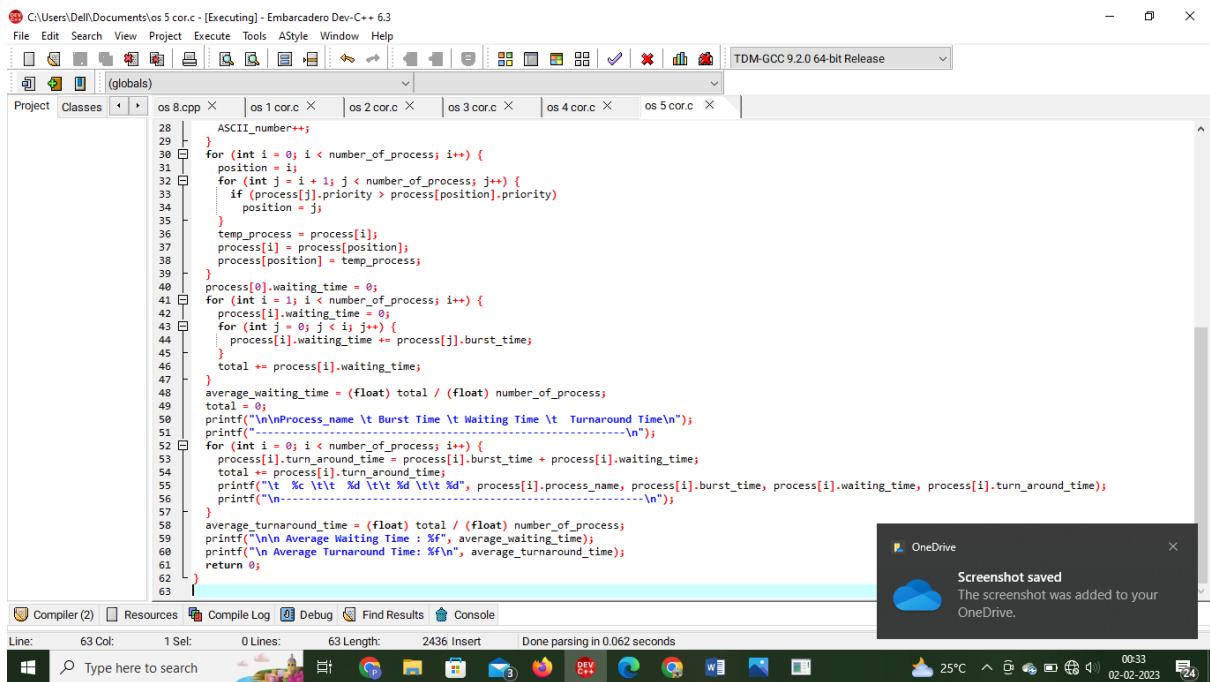
}

```

Input:



```
1 #include<stdio.h>
2 struct priority_scheduling {
3     char process_name;
4     int burst_time;
5     int waiting_time;
6     int turn_around_time;
7     int priority;
8 };
9 int main() {
10     int number_of_process;
11     int total = 0;
12     struct priority_scheduling temp_process;
13     int ASCII_number = 65;
14     int position;
15     float average_waiting_time;
16     float average_turnaround_time;
17     printf("Enter the total number of Processes: ");
18     scanf("%d", & number_of_process);
19     struct priority_scheduling process[number_of_process];
20     printf("\nPlease Enter the Burst Time and Priority of each process:\n");
21     for (int i = 0; i < number_of_process; i++) {
22         process[i].process_name = (char) ASCII_number;
23         printf("\nEnter the details of the process %c \n", process[i].process_name);
24         printf("Enter the burst time: ");
25         scanf("%d", & process[i].burst_time);
26         printf("Enter the priority: ");
27         scanf("%d", & process[i].priority);
28         ASCII_number++;
29     }
30     for (int i = 0; i < number_of_process; i++) {
31         position = i;
32         for (int j = i + 1; j < number_of_process; j++) {
33             if (process[j].priority > process[position].priority)
34                 position = j;
35         }
36         temp_process = process[i];
```



```
28         ASCII_number++;
29     }
30     for (int i = 0; i < number_of_process; i++) {
31         position = i;
32         for (int j = i + 1; j < number_of_process; j++) {
33             if (process[j].priority > process[position].priority)
34                 position = j;
35         }
36         temp_process = process[i];
37         process[i] = process[position];
38         process[position] = temp_process;
39     }
40     process[0].waiting_time = 0;
41     for (int i = 1; i < number_of_process; i++) {
42         process[i].waiting_time = 0;
43         for (int j = 0; j < i; j++) {
44             process[i].waiting_time += process[j].burst_time;
45         }
46         total += process[i].waiting_time;
47     }
48     average_waiting_time = (float) total / (float) number_of_process;
49     total = 0;
50     printf("\n\nProcess_name \t Burst Time \t Waiting Time \t Turnaround Time\n");
51     printf("-----\n");
52     for (int i = 0; i < number_of_process; i++) {
53         process[i].turn_around_time = process[i].burst_time + process[i].waiting_time;
54         total += process[i].turn_around_time;
55         printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process_name, process[i].burst_time, process[i].waiting_time, process[i].turn_around_time);
56         printf("\n-----\n");
57     }
58     average_turnaround_time = (float) total / (float) number_of_process;
59     printf("\n\n Average Waiting Time : %f", average_waiting_time);
60     printf("\n\n Average Turnaround Time: %f\n", average_turnaround_time);
61     return 0;
62 }
63 }
```

Output:

```
C:\Users\Deell\Documents\os 5 cor.exe
Enter the total number of Processes: 5
Please Enter the Burst Time and Priority of each process:
Enter the details of the process A
Enter the burst time: 10
Enter the priority: 3
Enter the details of the process B
Enter the burst time: 1
Enter the priority: 1
Enter the details of the process C
Enter the burst time: 2
Enter the priority: 4
Enter the details of the process D
Enter the burst time: 1
Enter the priority: 5
Enter the details of the process E
Enter the burst time: 5
Enter the priority: 2

Process_name    Burst Time    Waiting Time    Turnaround Time
-----
D              1              0              1
C              2              1              3
A             10              3             13
E              5             13             18
B              1             18             19
-----

Average Waiting Time : 7.000000
Average Turnaround Time: 10.800000

Process exited after 15.36 seconds with return value 0
Press any key to continue . . .
```

6. Construct a C program to implement pre-emptive priority scheduling algorithm.

Program:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int burst_time[20],p[20],waiting_time[20],tat[20],i,j,n,total=0,pos,temp;
```

```
float avg_waiting_time,avg_tat;
```

```
printf("please enter number of process: ");
```

```
scanf("%d",&n);
```

```
printf("\n enter the Burst Time:\n");
```

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

```
{
```

```
printf("p%d:",i+1);
```

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

```
p[i]=i+1;
```

```

}
for(i=0;i<n;i++)
{
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(burst_time[j]<burst_time[pos])
            pos=j;
    }
    temp=burst_time[i];
    burst_time[i]=burst_time[pos];
    burst_time[pos]=temp;
    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}
waiting_time[0]=0;
for(i=1;i<n;i++)
{
    waiting_time[i]=0;
    for(j=0;j<i;j++)
        waiting_time[i]+=burst_time[j];
    total+=waiting_time[i];
}
avg_waiting_time=(float)total/n;
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{

```

```

    tat[i]=burst_time[i]+waiting_time[i];

    total+=tat[i];

    printf("\np%d\t\t %d\t\t %d\t\t\t %d",p[i],burst_time[i],waiting_time[i],tat[i]);
}

avg_tat=(float)total/n;

printf("\n\n the average Waiting Time=%f",avg_waiting_time);

printf("\n the average Turnaround Time=%f\n",avg_tat);
}

```

Input:

```

cor.c - [Executing] - Embarcadero Dev-C++ 6.3
ct  Execute  Tools  AStyle  Window  Help

1 cor.c x | os 2 cor.c x | os 3 cor.c x | os 4 cor.c x | os 5 cor.c x | os 8 cor.c x | os 7 cor.c x | [x] os 6 cor.c x

#include<stdio.h>
int main()
{
    int burst_time[20],p[20],waiting_time[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_waiting_time,avg_tat;
    printf("please enter number of process: ");
    scanf("%d",&n);
    printf("\n enter the Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&burst_time[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(burst_time[j]<burst_time[pos])
                pos=j;
        }
        temp=burst_time[i];
        burst_time[i]=burst_time[pos];
        burst_time[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }
    waiting_time[0]=0;

    temp=burst_time[i];
    burst_time[i]=burst_time[pos];
    burst_time[pos]=temp;
    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}
waiting_time[0]=0;
for(i=1;i<n;i++)
{
    waiting_time[i]=0;
    for(j=0;j<i;j++)
        waiting_time[i]+=burst_time[j];
    total+=waiting_time[i];
}
avg_waiting_time=(float)total/n;
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
    tat[i]=burst_time[i]+waiting_time[i];
    total+=tat[i];
    printf("\np%d\t\t %d\t\t %d\t\t\t %d",p[i],burst_time[i],waiting_time[i],tat[i]);
}
avg_tat=(float)total/n;
printf("\n\n the average Waiting Time=%f",avg_waiting_time);
printf("\n the average Turnaround Time=%f\n",avg_tat);
}

```

Output:

```
please enter number of process: 5
enter the Burst Time:
p1:6
p2:2
p3:8
p4:3
p5:4
Process   Burst Time   Waiting Time   Turnaround Time
p2        2         0             2
p4        3         2             5
p5        4         5             9
p1        6         9            15
p3        8        15            23
the average Waiting Time=6.200000
the average Turnaround Time=10.800000
-----
Process exited after 14.14 seconds with return value 0
Press any key to continue . . .
```

7. Construct a C program to implement non-preemptive SJF algorithm.

Program:

```
#include<stdio.h>
```

```
int main() {
    int time, burst_time[10], at[10], sum_burst_time = 0, smallest, n, i;
    int sumt = 0, sumw = 0;
    printf("enter the no of processes : ");
    scanf("%d", & n);
    for (i = 0; i < n; i++) {
        printf("the arrival time for process P%d : ", i + 1);
        scanf("%d", & at[i]);
        printf("the burst time for process P%d : ", i + 1);
        scanf("%d", & burst_time[i]);
        sum_burst_time += burst_time[i];
    }
    burst_time[9] = 9999;
    for (time = 0; time < sum_burst_time;) {
        smallest = 9;
        for (i = 0; i < n; i++) {
```



```

    if (at[i] <= time && burst_time[i] > 0 && burst_time[i] < burst_time[smallest])

        smallest = i;
}

printf("P[%d]\t|\t%d\t|\t%d\n", smallest + 1, time + burst_time[smallest] -
at[smallest], time - at[smallest]);

sumt += time + burst_time[smallest] - at[smallest];

sumw += time - at[smallest];

time += burst_time[smallest];

burst_time[smallest] = 0;
}

printf("\n\n average waiting time = %f", sumw * 1.0 / n);

printf("\n\n average turnaround time = %f", sumt * 1.0 / n);

return 0;
}

```

Input:

```

1  #include<stdio.h>
2
3  int main() {
4      int time, burst_time[10], at[10], sum_burst_time = 0, smallest, n, i;
5      int sumt = 0, sumw = 0;
6      printf("enter the no of processes : ");
7      scanf("%d", &n);
8      for (i = 0; i < n; i++) {
9          printf("the arrival time for process P%d : ", i + 1);
10         scanf("%d", &at[i]);
11         printf("the burst time for process P%d : ", i + 1);
12         scanf("%d", &burst_time[i]);
13         sum_burst_time += burst_time[i];
14     }
15     burst_time[9] = 9999;
16     for (time = 0; time < sum_burst_time; ) {
17         smallest = 9;
18         for (i = 0; i < n; i++) {
19             if (at[i] <= time && burst_time[i] > 0 && burst_time[i] < burst_time[smallest])
20                 smallest = i;
21         }
22         printf("P[%d]\t|\t%d\t|\t%d\n", smallest + 1, time + burst_time[smallest] - at[smallest], time - at[smallest]);
23         sumt += time + burst_time[smallest] - at[smallest];
24         sumw += time - at[smallest];
25         time += burst_time[smallest];
26         burst_time[smallest] = 0;
27     }
28     printf("\n\n average waiting time = %f", sumw * 1.0 / n);
29     printf("\n\n average turnaround time = %f", sumt * 1.0 / n);
30     return 0;
31 }
32
33

```

Output:

```
C:\Users\ DELL\Documents>os 7 cor.exe
enter the no of processes : 5
the arrival time for process P1 : 2
the burst time for process P1 : 6
the arrival time for process P2 : 5
the burst time for process P2 : 2
the arrival time for process P3 : 1
the burst time for process P3 : 8
the arrival time for process P4 : 0
the burst time for process P4 : 3
the arrival time for process P5 : 4
the burst time for process P5 : 4
P[4] | 3 | 0
P[1] | 7 | 1
P[2] | 6 | 4
P[5] | 11 | 7
P[3] | 22 | 14

average waiting time = 5.200000
average turnaround time = 9.800000
-----
Process exited after 31.7 seconds with return value 0
Press any key to continue . . .
```

8. Construct a C program to simulate Round Robin scheduling algorithm with C.

Program:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
    int i, NOP, sum=0, count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
```

```
    float avg_wt, avg_tat;
```

```
    printf(" Total number of process in the system: ");
```

```
    scanf("%d", &NOP);
```

```
    y = NOP;
```

```
    for(i=0; i<NOP; i++)
```

```
    {
```

```
        printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
```

```
        printf(" Arrival time is: \t");
```

```

scanf("%d", &at[i]);
printf(" \nBurst time is: \t");
scanf("%d", &bt[i]);
temp[i] = bt[i];
}
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0)
{
sum = sum + temp[i];
temp[i] = 0;
count=1;
}
else if(temp[i] > 0)
{
temp[i] = temp[i] - quant;
sum = sum + quant;
}
if(temp[i]==0 && count==1)
{
y--;
printf("\nProcess No[%d] \t\t %d\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);

wt = wt+sum-at[i]-bt[i];
tat = tat+sum-at[i];
count =0;
}
}

```

```

    if(i==NOP-1)
    {
        i=0;
    }

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

    {
        i++;
    }

    else

    {
        i=0;
    }

} avg_wt = wt * 1.0/NOP;

avg_tat = tat * 1.0/NOP;

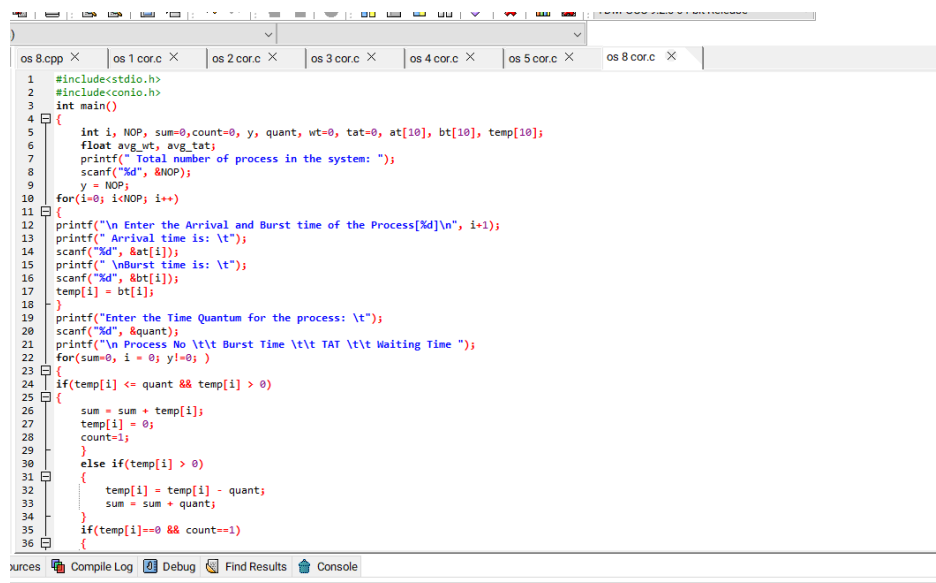
printf("\n Average Turn Around Time: \t%f", avg_wt);

printf("\n Average Waiting Time: \t%f", avg_tat);

getch();}

Input:

```



```

1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5      int i, NOP, sum=0, count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
6      float avg_wt, avg_tat;
7      printf(" Total number of process in the system: ");
8      scanf("%d", &NOP);
9      y = NOP;
10     for(i=0; i<NOP; i++)
11     {
12         printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
13         printf(" Arrival time is: \t");
14         scanf("%d", &at[i]);
15         printf("\nBurst time is: \t");
16         scanf("%d", &bt[i]);
17         temp[i] = bt[i];
18     }
19     printf("Enter the Time Quantum for the process: \t");
20     scanf("%d", &quant);
21     printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
22     for(sum=0, i = 0; y!=0; )
23     {
24         if(temp[i] <= quant && temp[i] > 0)
25         {
26             sum = sum + temp[i];
27             temp[i] = 0;
28             count=i;
29         }
30         else if(temp[i] > 0)
31         {
32             temp[i] = temp[i] - quant;
33             sum = sum + quant;
34         }
35         if(temp[i]==0 && count==1)
36         {

```

```
27 temp[i] = 0;
28 count=1;
29 }
30 else if(temp[i] > 0)
31 {
32     temp[i] = temp[i] - quant;
33     sum = sum + quant;
34 }
35 if(temp[i]==0 && count==1)
36 {
37     y--;
38     printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
39     wt = wt+sum-at[i]-bt[i];
40     tat = tat+sum-at[i];
41     count =0;
42 }
43 if(i==NOP-1)
44 {
45     i=0;
46 }
47 else if(at[i+1]<=sum)
48 {
49     i++;
50 }
51 else
52 {
53     i=0;
54 }
55 }
56 avg_wt = wt * 1.0/NOP;
57 avg_tat = tat * 1.0/NOP;
58 printf("\n Average Turn Around Time: \t%f", avg_wt);
59 printf("\n Average Waiting Time: \t%f", avg_tat);
60 getch();
61 }
62 }
```

Output:

```
C:\Users\De\Documents\os 8 cor.exe
Enter the Arrival and Burst time of the Process[3]
Arrival time is: 2
Burst time is: 1
Enter the Arrival and Burst time of the Process[4]
Arrival time is: 3
Burst time is: 4
Enter the Arrival and Burst time of the Process[5]
Arrival time is: 4
Burst time is: 5
Enter the Arrival and Burst time of the Process[6]
Arrival time is: 5
Burst time is: 2
Enter the Time Quantum for the process: 1

Process No      Burst Time      TAT      Waiting Time
Process No[3]   1               1         0
Process No[2]   2               7         5
Process No[6]   2               6         4
Process No[1]   3               12        9
Process No[4]   4               12        8
Process No[5]   5               13        8
Average Turn Around Time: 5.666667
Average Waiting Time: 8.500000
```