Assignment

No.

1

Course: Operating System

Lab

Section:

106354

Assigned By: UBAID

ULLAH



**LAB**

**ASSIGNMENT**

**#.**

**1**

**Course:**

Operating System

Assigned By:

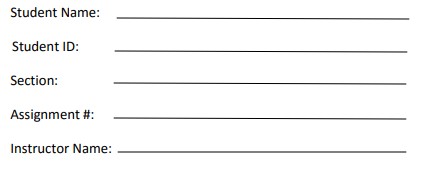
UBAID ULLAH

**SUMMER**

**2021**

**Points To Remember**

1. **Total Marks Of Assignment is 10.**
2. **If the code/answer is found copied, straight zero will be marked.**
3. **Straight zero will be marked for incorrect submissions.**
4. **Submit Your Assignment within due date.**
5. **Late submissions will not be considered.**
6. **Assignment must only be submitted on LMS. Submission on any other platform will not be accepted.**



1. **Your file name should be like**

**(OS\_Assigment#1\_StudentName\_StudentID\_Section)**

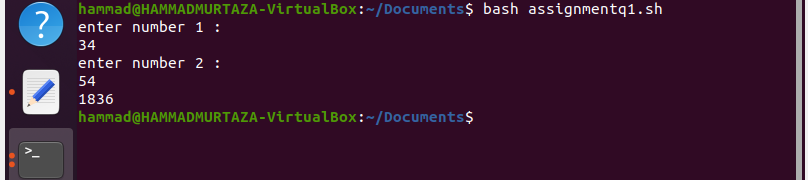
1. **File submitted answering to this assignment must contain:** 
   * 1. Code of the program along with the screen shots of the output.
     2. All work must be in a proper, quality, neat and clean way as with all work in this course, 10% of the grade is for quality of presentation.
2. **Output must be in the formatted form.**
3. **Cover page of your file must contain:**

**HAMMAD MURTAZA**

**Question No.1**

**Write a shell script that multiplies two numbers without using the mathematical operator (\*).** (Marks=2.5)





**Question No. 2**

**Write a C program which dynamically allocates the memory to the pointer variable and inputs the following values from user:**

* 1. (30, 4, 38, 18, 22, 44, 210) in the allocated memory and print on the screen.

* 1. Re-Allocate the memory to the pointer variable and now input the following values from user in the re-allocated memory (44, 14, 2, 4, 40, 30) and print them.

* 1. Add check the same values occurring before and after the re-allocation of the memory and store them in an array and then print the array on the screen.
  2. Also Count the same values and print the total number of same values are e.g

(Marks=2.5)

Code:

#include <stdio.h>

#include <stdlib.h>

int main()

{

int \*ptr, i , n1, n2;

printf("Enter size of Number: ");

scanf("%d", &n1);

ptr = (int\*) malloc(n1 \* sizeof(int));

printf("Enter Number of Elements: ");

for(i = 0; i < n1; ++i)

scanf("%d",ptr + i);

printf("Array of Element you enter: ");

for(i = 0; i<n1; i++)

printf("%d ",\*(ptr+i));

printf("\nEnter the new size: ");

scanf("%d", &n2);

// rellocating the memory

ptr = realloc(ptr, n2 \* sizeof(int));

printf("Enter New Number of Elements: ");

for(i = 0; i < n2; ++i)

scanf("%d", ptr + i);

printf("Array of New Elements you enter: ");

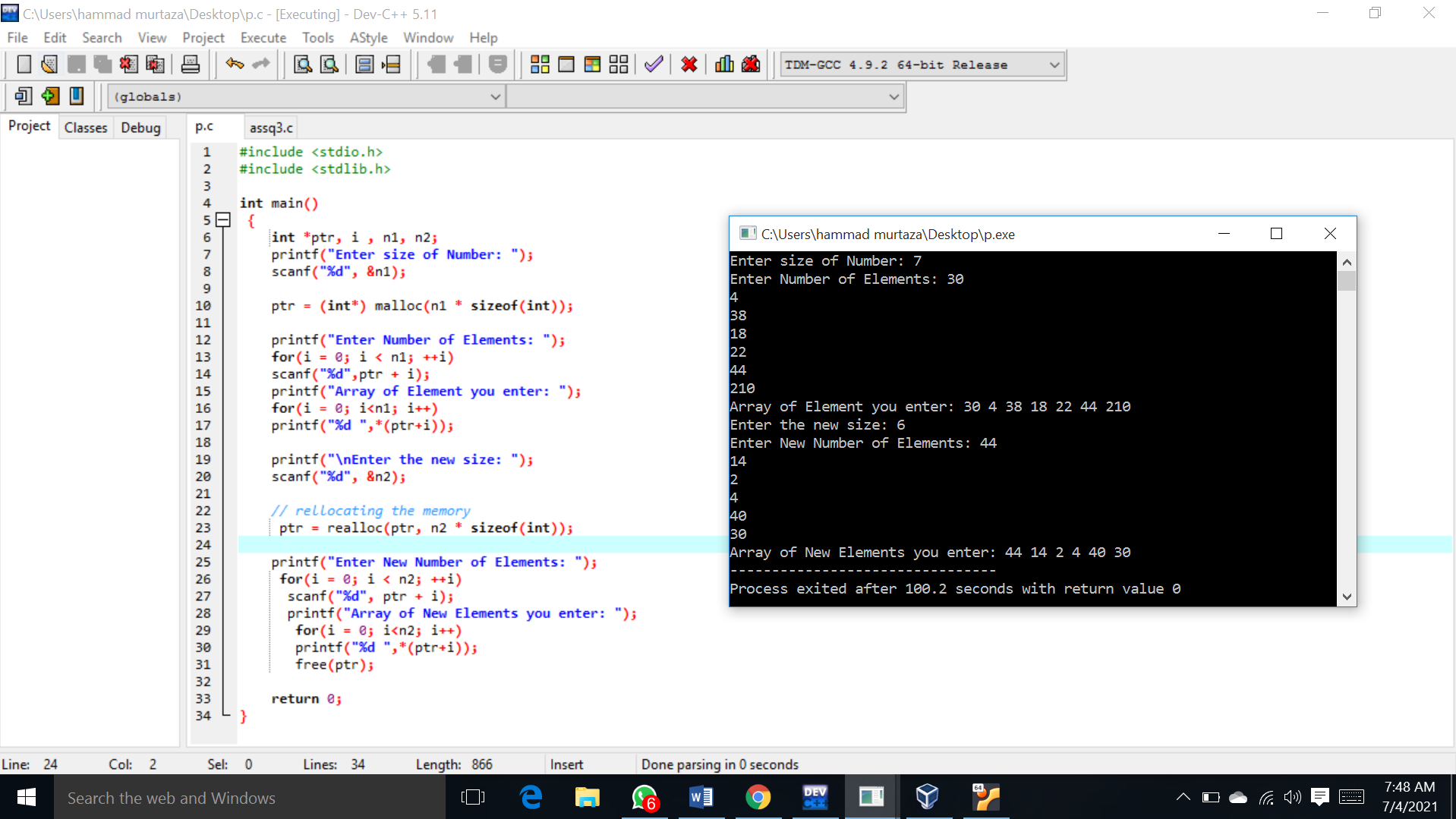
for(i = 0; i<n2; i++)

printf("%d ",\*(ptr+i));

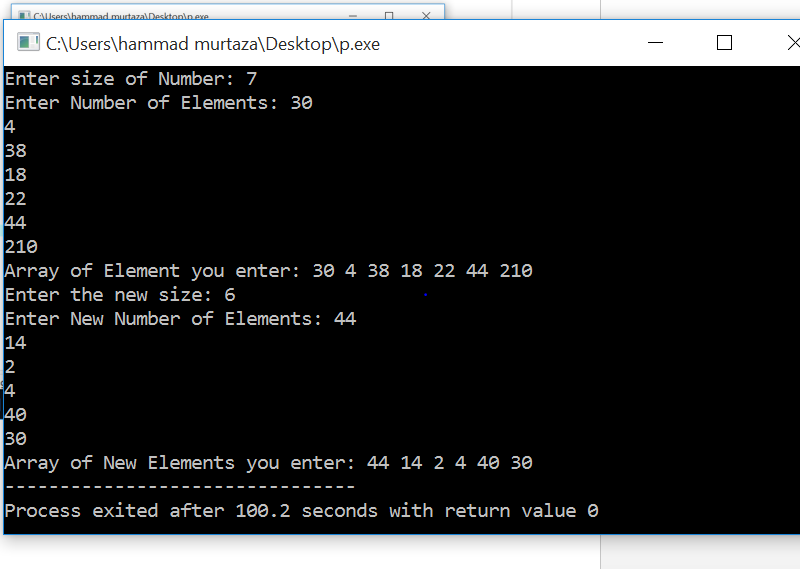
free(ptr);

return 0;

}



Output:



**Question No. 3**

**Implement the following scheduling algorithms using C Language and must use**

**the given Priorities and Burst Time:** (Marks=5)

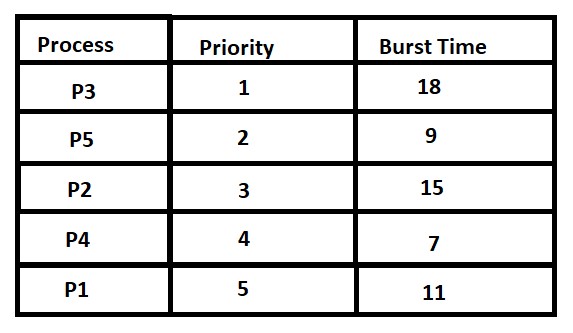
1. **First Come First Serve**
2. **Round Robin**
3. Key points to remember
   1. Your program must take input only once for the both algorithms from user.
   2. Then print the table for both algorithms, output table must contain the following.
      * **1. Process ID, 2. Priority, 3. Burst Time, 4. Waiting Time, 5. Turnaround Time, 6. Avg Waiting\_Time,**

**7. Avg Turnaround\_Time.**

* + - And then compare the **Avg Waiting\_Time** and **Avg**

**Turnaround\_Time** at the end. And give your analysis which one is better according to their **Avg Waiting\_Time** and **Avg Turnaround\_Time.**

* + - Use the following chart for Priorities and Burst Time for Processes.



Code:

#include<stdio.h>

#include<string.h>

struct Process{

int pid;

int bt;

int wt;

int tt;

int pr;

};

void charac(char ch,int n)

{

int i;

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

printf("%c", ch);

printf("\n");

}

void graphics(const char str[])

{

int i;

int len = (80-strlen(str))/2;

charac('\*',80);

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

printf("%c", ' ');

printf("%s\n", str);

charac('\*',80);

}

int main()

{

int n,i,j,TotW,TotT, ts, tt, wt, check;

float AvgW,AvgT, FAW, FAT, RAW, RAT;

graphics("Sample output");

//printf("Enter the no. of Process : ");

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

n = 5;

struct Process p[n];

struct Process p2[n];

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

{

p[i].pid = i+1;

printf("Priority and Brust Time for P%d : ", p[i].pid);

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

p2[i].pid = p[i].pid;

p2[i].pr = p[i].pr;

p2[i].bt = p[i].bt;

}

printf("\n");

printf("Enter Time Slice for Round Robin Scheduling ");

scanf("%d", &ts);

printf("\n");

for(i=0;i<n-1;i++)

for(j=0;j<n-i-1;j++)

if(p2[j].pr > p2[j+1].pr)

{

struct Process temp;

temp = p2[j];

p2[j] = p2[j+1];

p2[j+1] = temp;

}

TotW = 0;

TotT = 0;

tt=0;

wt=0;

while(1)

{

check = 0;

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

{

if(p2[i].bt > 0)

{

if(p2[i].bt <= ts)

{

p2[i].wt = tt;

p2[i].tt = p2[i].bt + tt;

p2[i].bt = 0;

}

else

{

p2[i].wt = tt;

p2[i].tt = tt + ts;

p2[i].bt = p2[i].bt - ts;

}

wt = p2[i].wt;

tt = p2[i].tt;

TotW = TotW + wt;

TotT = TotT + tt;

}

if(p2[i].bt != 0)

check = 1;

}

if(check == 0)

break;

}

TotW = 0;

TotT = 0;

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

{

p2[i].bt = p[i].bt;

p2[i].wt = p2[i].tt - p2[i].bt;

TotW = TotW + p2[i].wt;

TotT = TotT + p2[i].tt;

}

RAW = AvgW = TotW \* 1.0 / n;

RAT = AvgT = TotT \* 1.0 / n;

graphics(" Calculation Round Robin Scheduling");

printf("%15s%15s%15s%15s%20s\n", "Process", "Priority", "Burst Time", "Waiting Time ", "Turnaround Time");

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

printf("%14s%d%15d%15d%15d%20d\n", "P", p2[i].pid, p2[i].pr, p2[i].bt, p2[i].wt, p2[i].tt);

printf("\n");

printf("avg Waiting Time using Round Robin: %f\n", AvgW);

printf("avg Turnaround Time using Round Robin : %f\n", AvgT);

printf("\n");

for(i=0;i<n-1;i++)

for(j=0;j<n-i-1;j++)

if(p[j].pr > p[j+1].pr)

{

struct Process temp;

temp = p[j];

p[j] = p[j+1];

p[j+1] = temp;

}

p[0].wt = 0;

p[0].tt = p[0].bt;

TotW = 0;

TotT = p[0].tt;

for(i=1;i<n;i++)

{

p[i].wt = p[i-1].tt;

p[i].tt = p[i].wt + p[i].bt;

TotW = TotW + p[i].wt;

TotT = TotT + p[i].tt;

}

FAW = AvgW = TotW \* 1.0 / n;

FAT = AvgT = TotT \* 1.0 / n;

graphics(" Calculation for FCFS Scheduling");

printf("%15s%15s%15s%15s%20s\n", "Process", "Priority", "Burst Time", "Waiting Time", "Turnaround Time");

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

printf("%14s%d%15d%15d%15d%20d\n", "P", p[i].pid, p[i].pr, p[i].bt, p[i].wt, p[i].tt);

printf("\n");

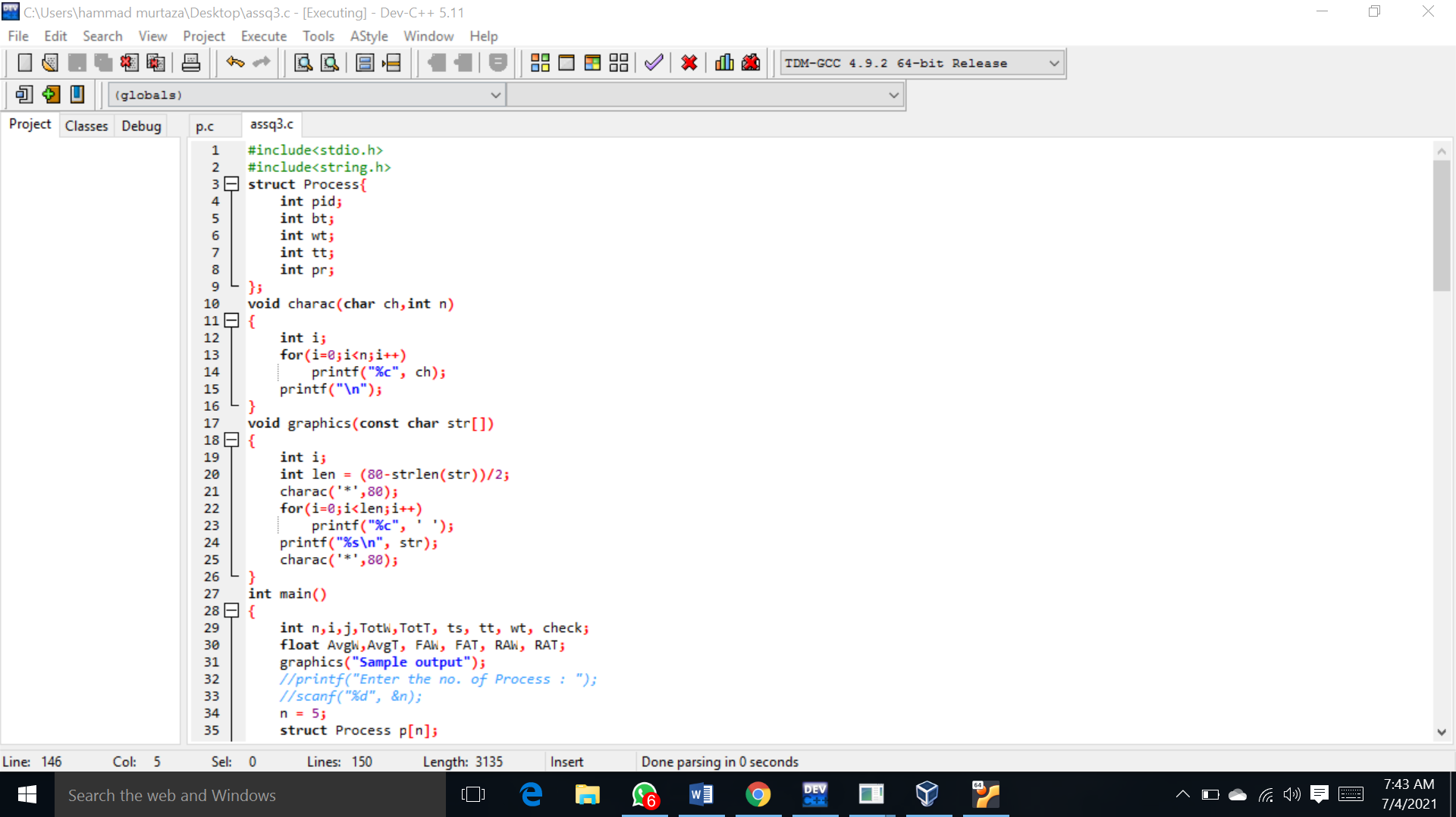
printf("avg WT FCFS : %f\n", AvgW);

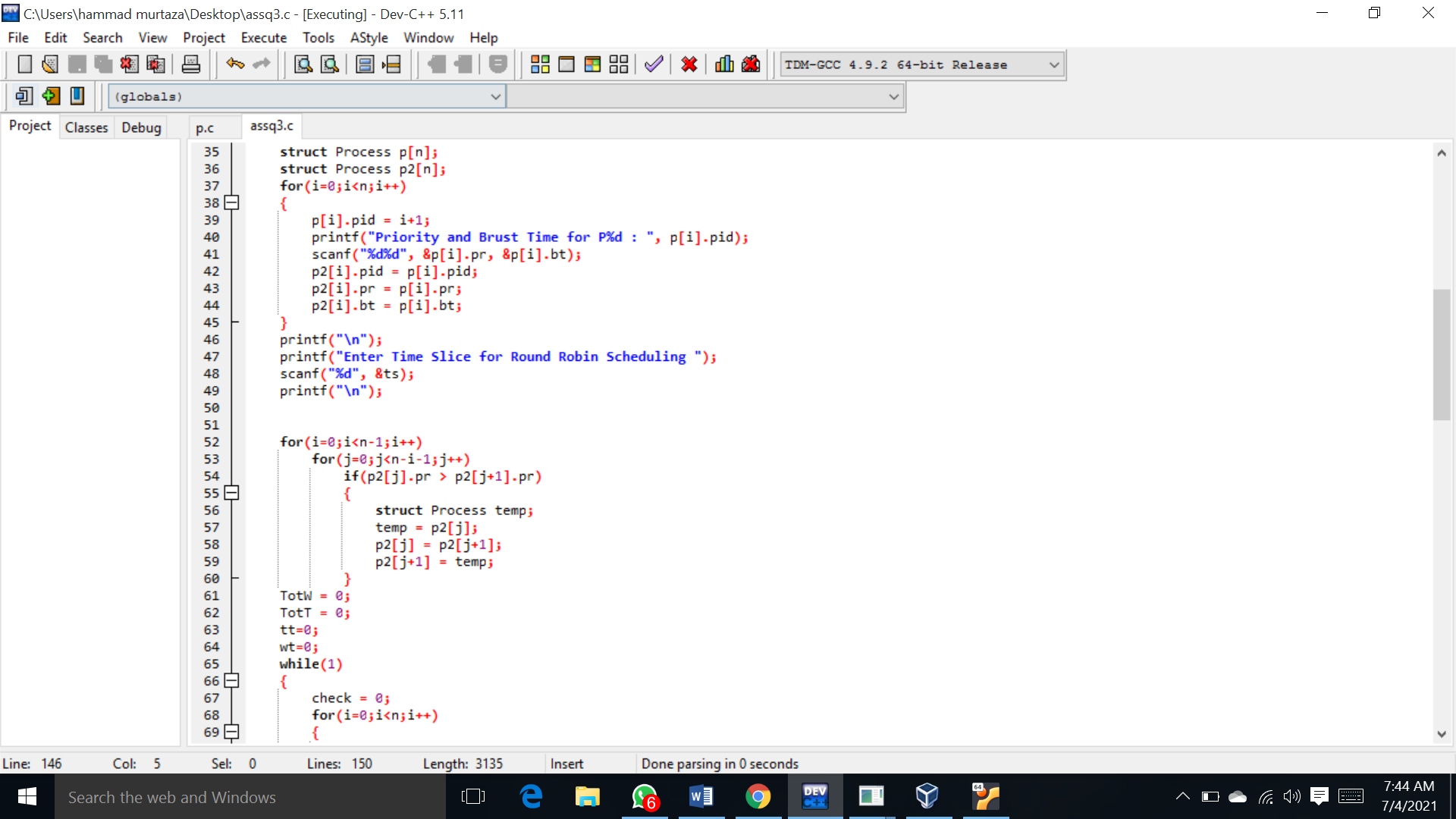
printf("avg TT FCFS : %f\n", AvgT);

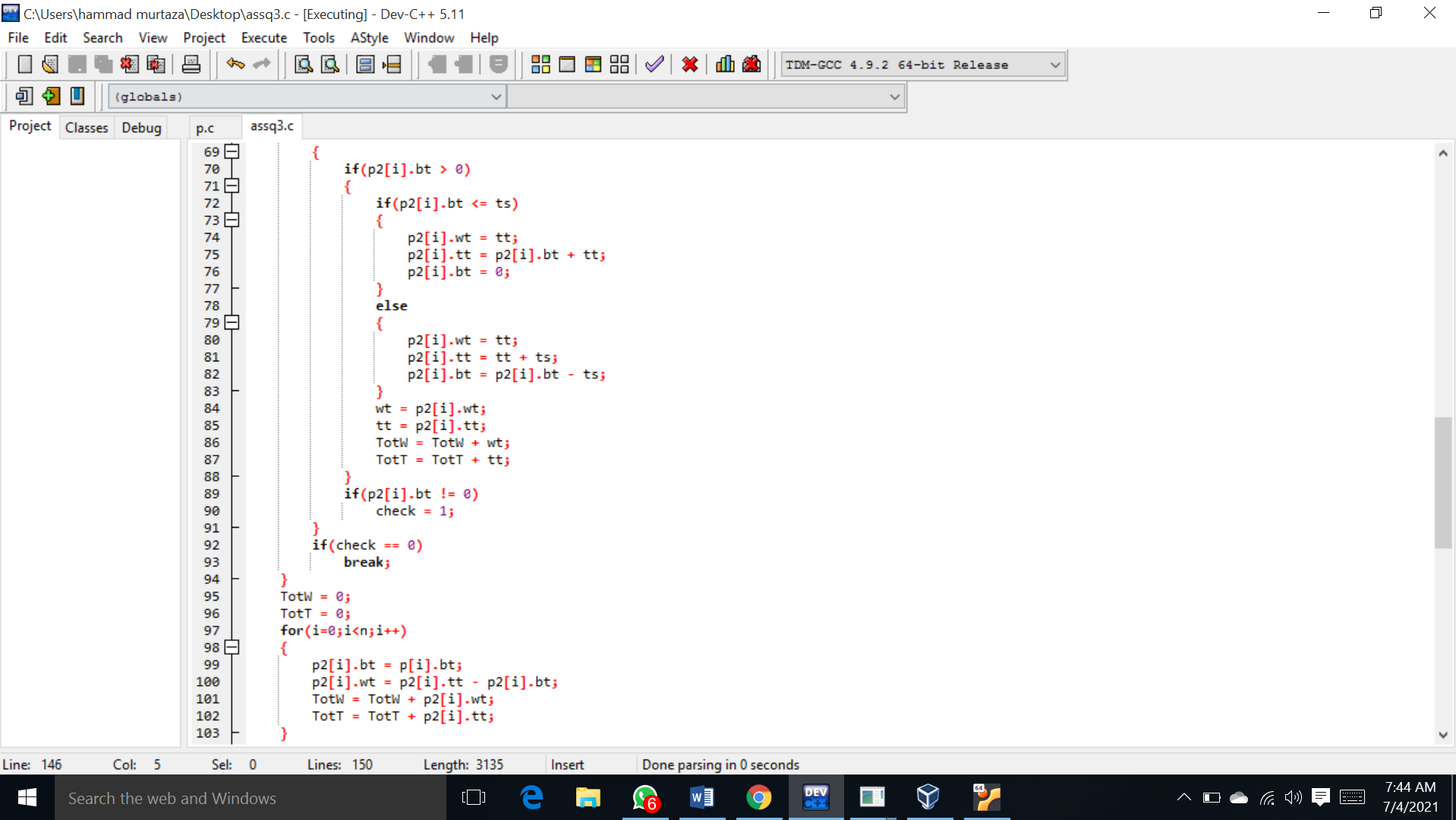
printf("\n");

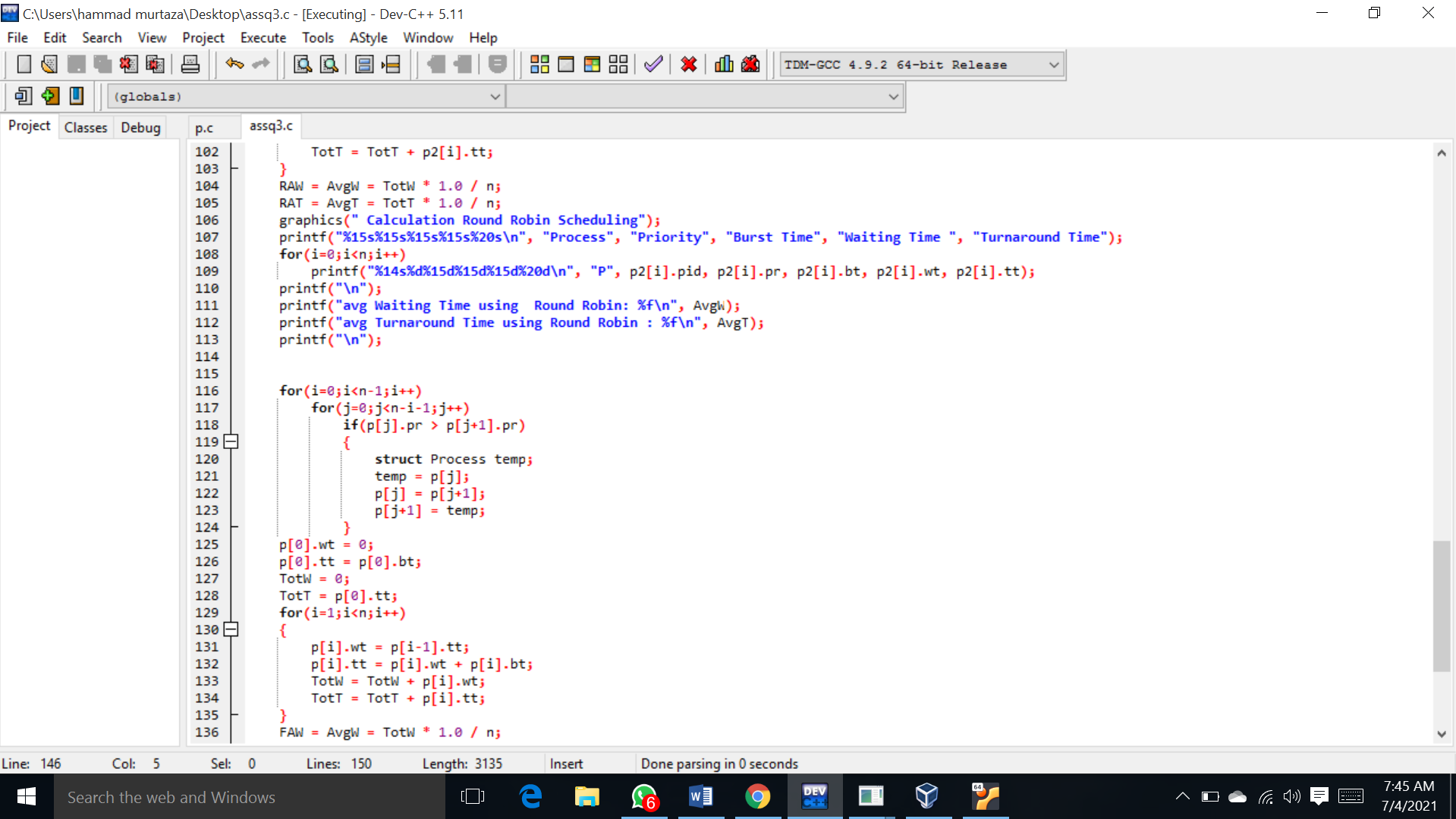
return 0;

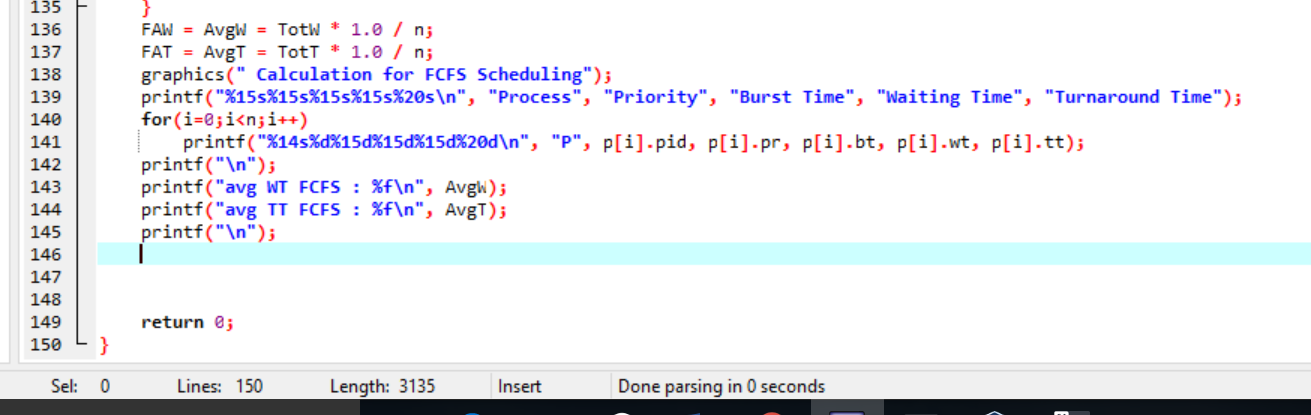
}



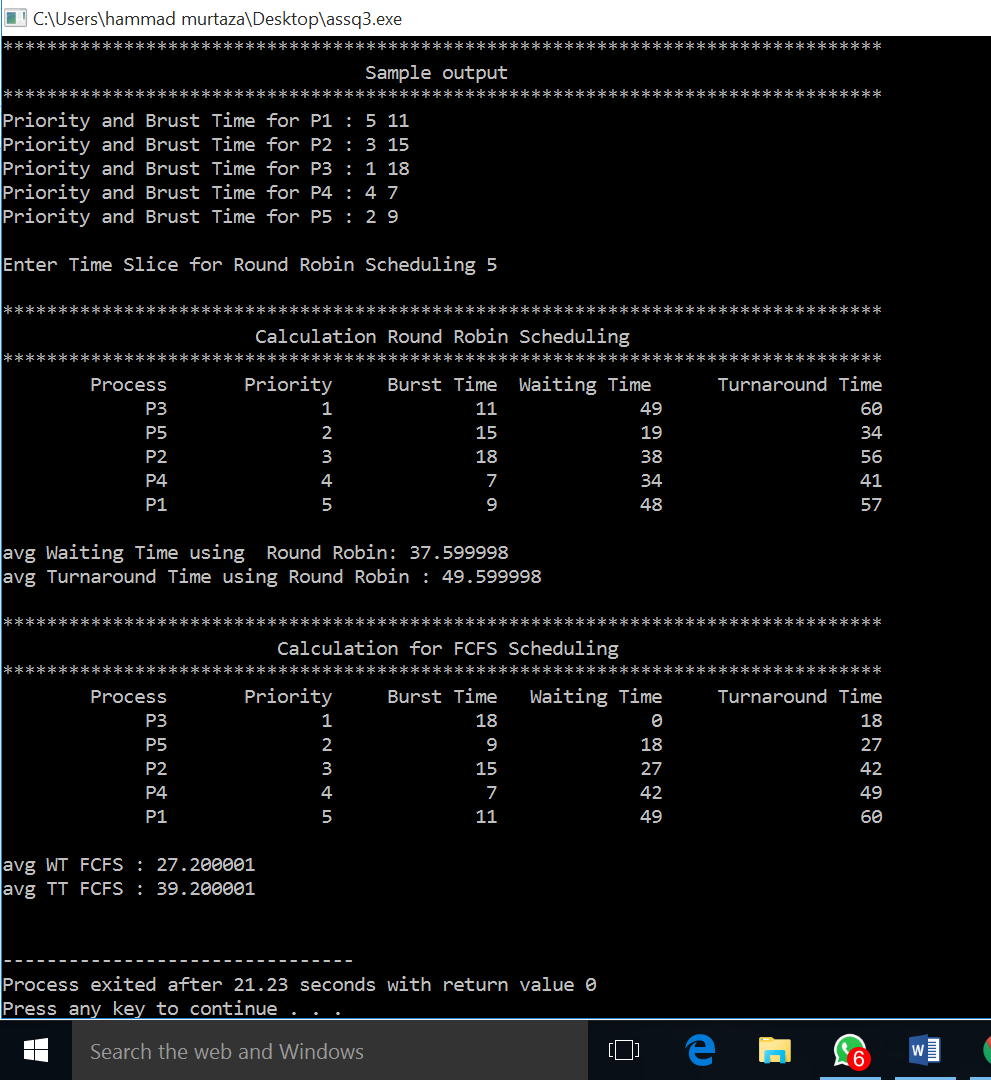


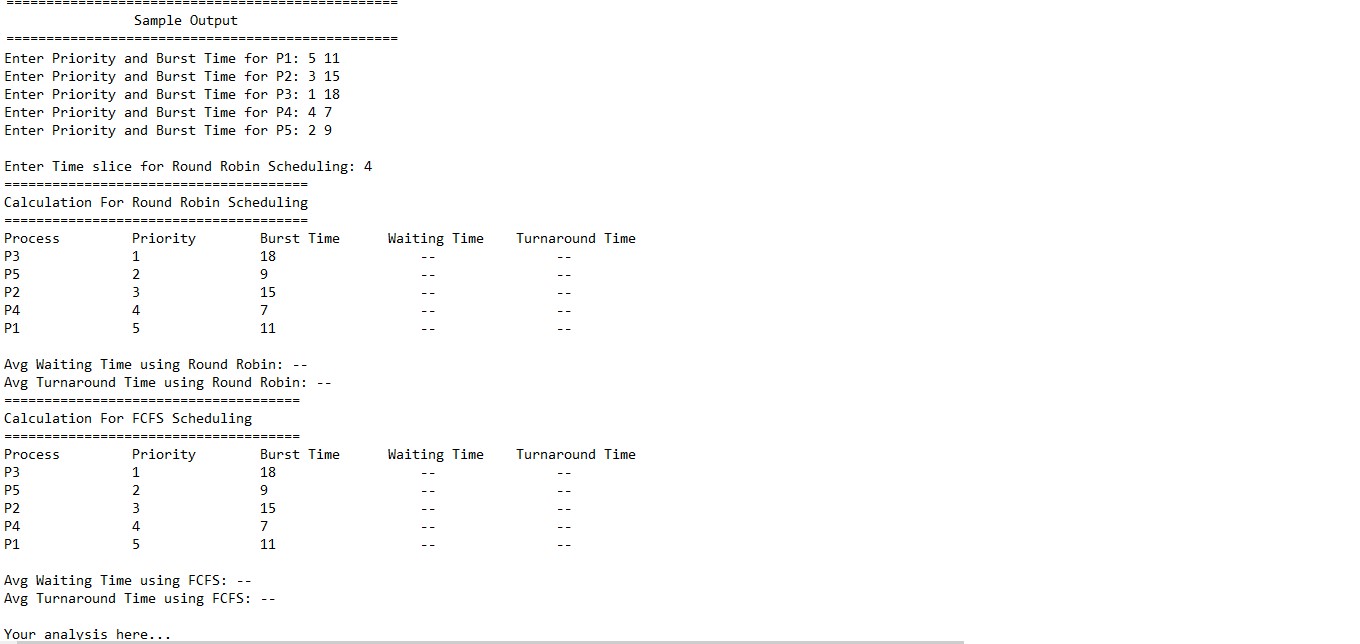






Output:





Best Of Luck