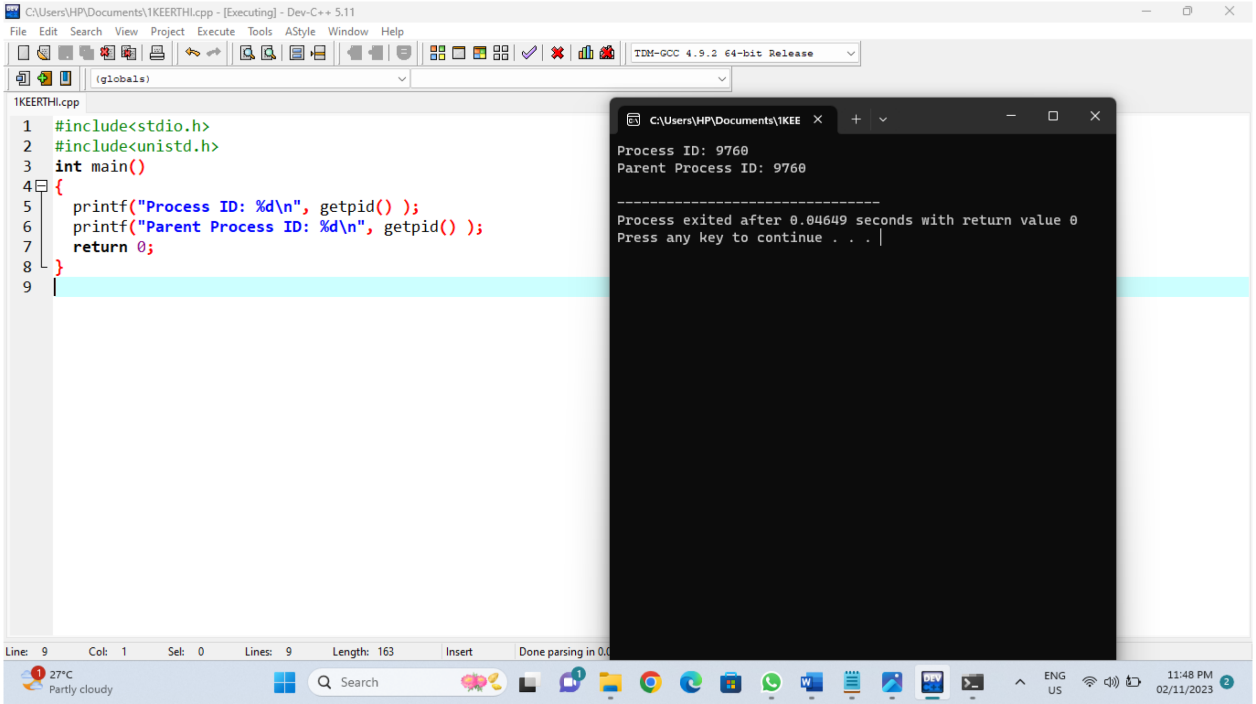
Practical programs

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

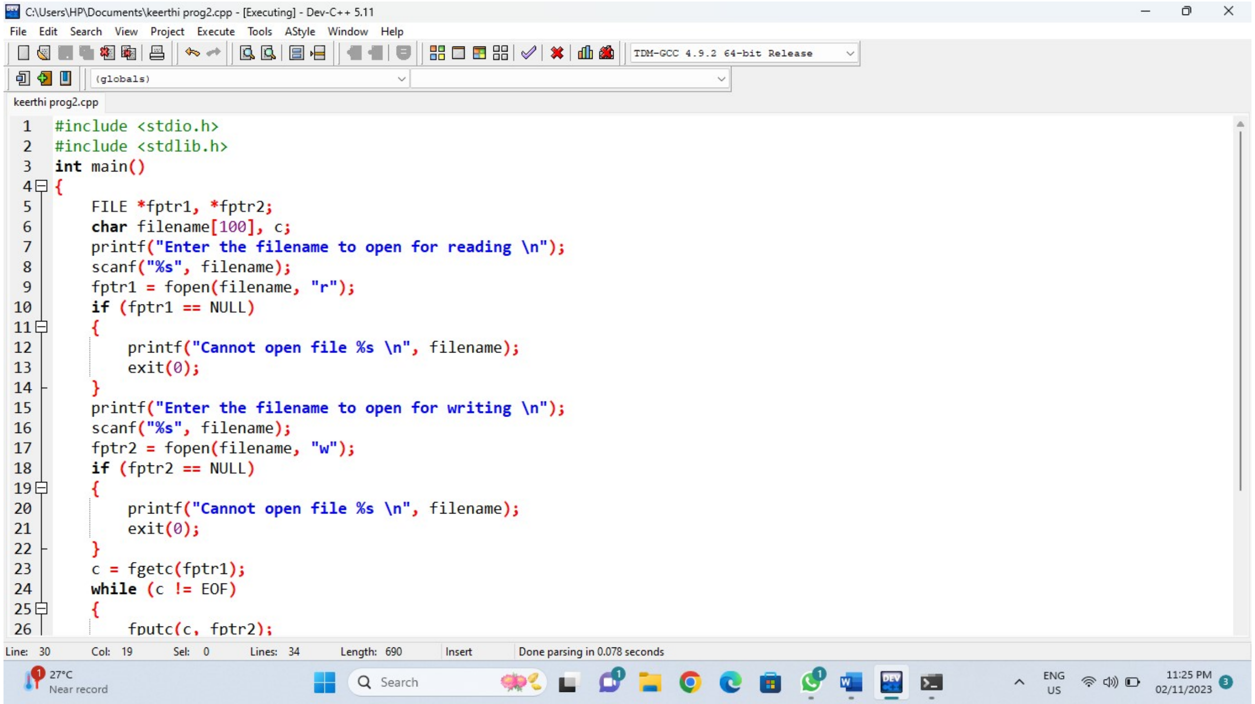

C:\Users\HP\Documents\1KEERTHI.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
1KEERTHI.cpp
C:\Users\HP\Documents\1KEE
Line: 9 Col: 1 Sel: 0 Lines: 9 Length: 163 Insert Done parsing in 0.0
27°C Partly cloudy
Search
ENG US
11:48 PM 02/11/2023

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

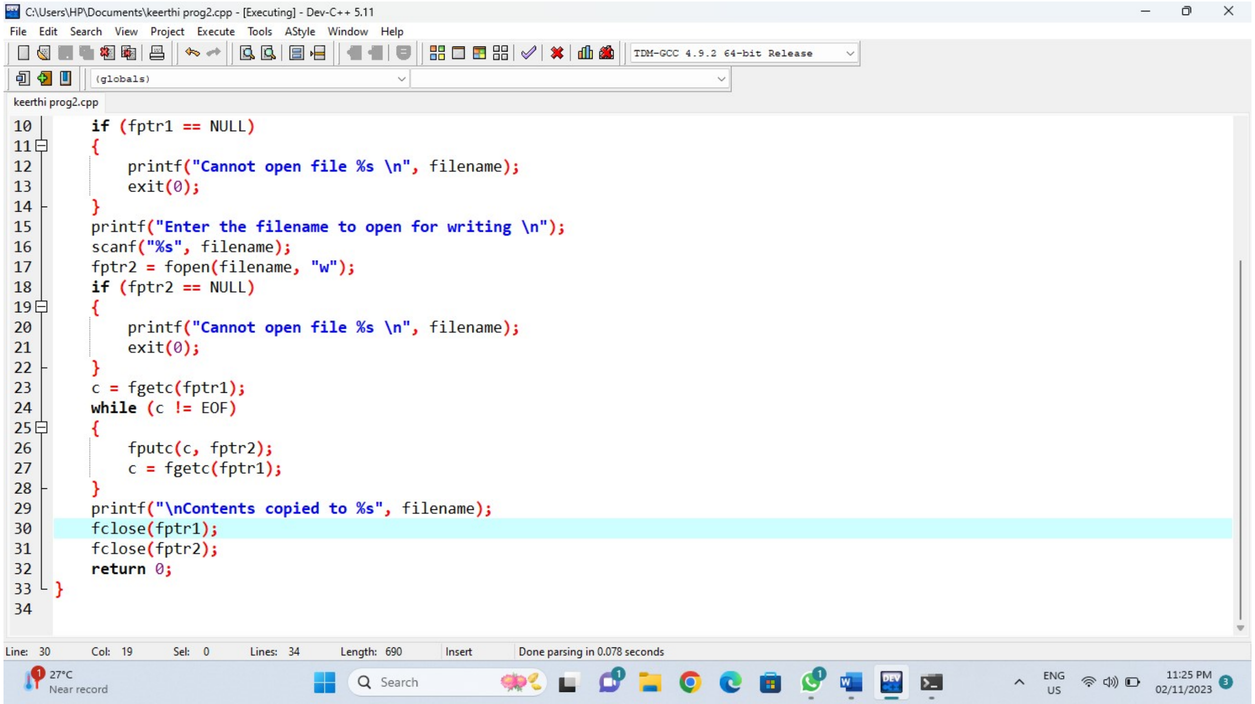
```
Process ID: 9760
Parent Process ID: 9760

--------------------------------
Process exited after 0.04649 seconds with return value 0
Press any key to continue . . .
```

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

Practical programs


C:\Users\HP\Documents\keerthi prog2.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
keerthi prog2.cpp
Line: 30 Col: 19 Sel: 0 Lines: 34 Length: 690 Insert Done parsing in 0.078 seconds
27°C Near record
Search
ENG US
11:25 PM 02/11/2023

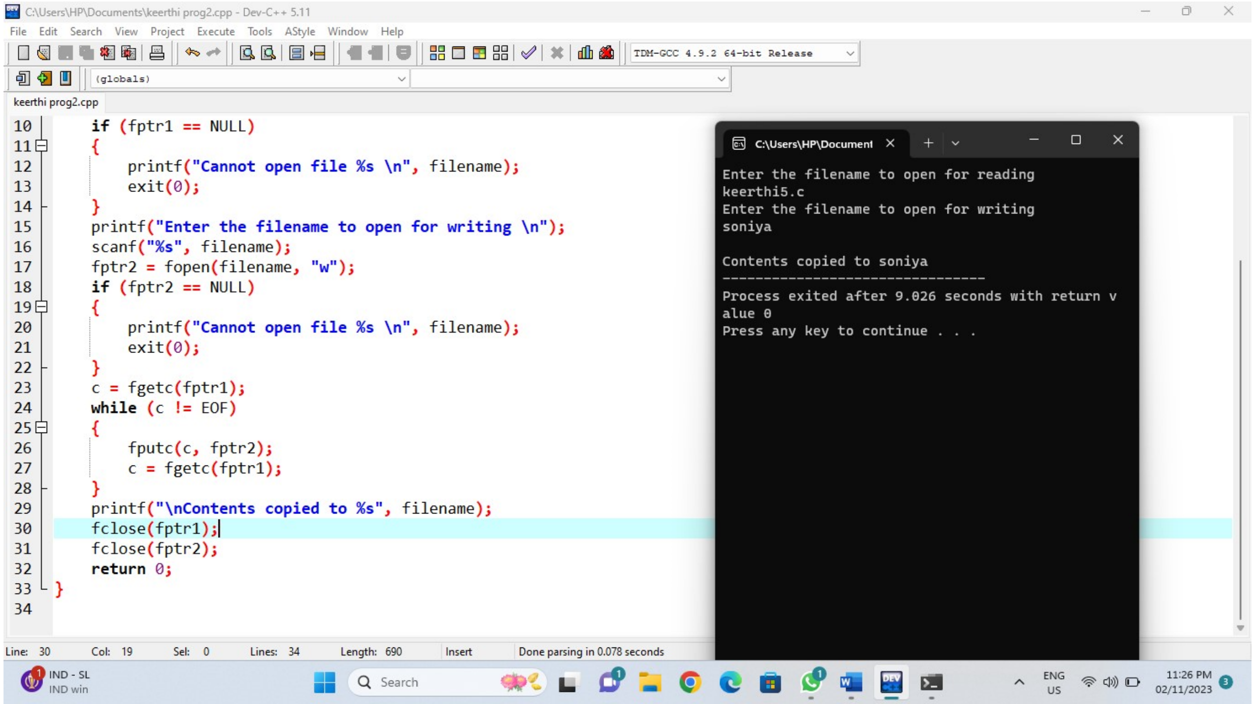
```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;
    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
```


C:\Users\HP\Documents\keerthi prog2.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
keerthi prog2.cpp
Line: 30 Col: 19 Sel: 0 Lines: 34 Length: 690 Insert Done parsing in 0.078 seconds
27°C Near record
Search
ENG US
11:25 PM 02/11/2023

```c
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}

```

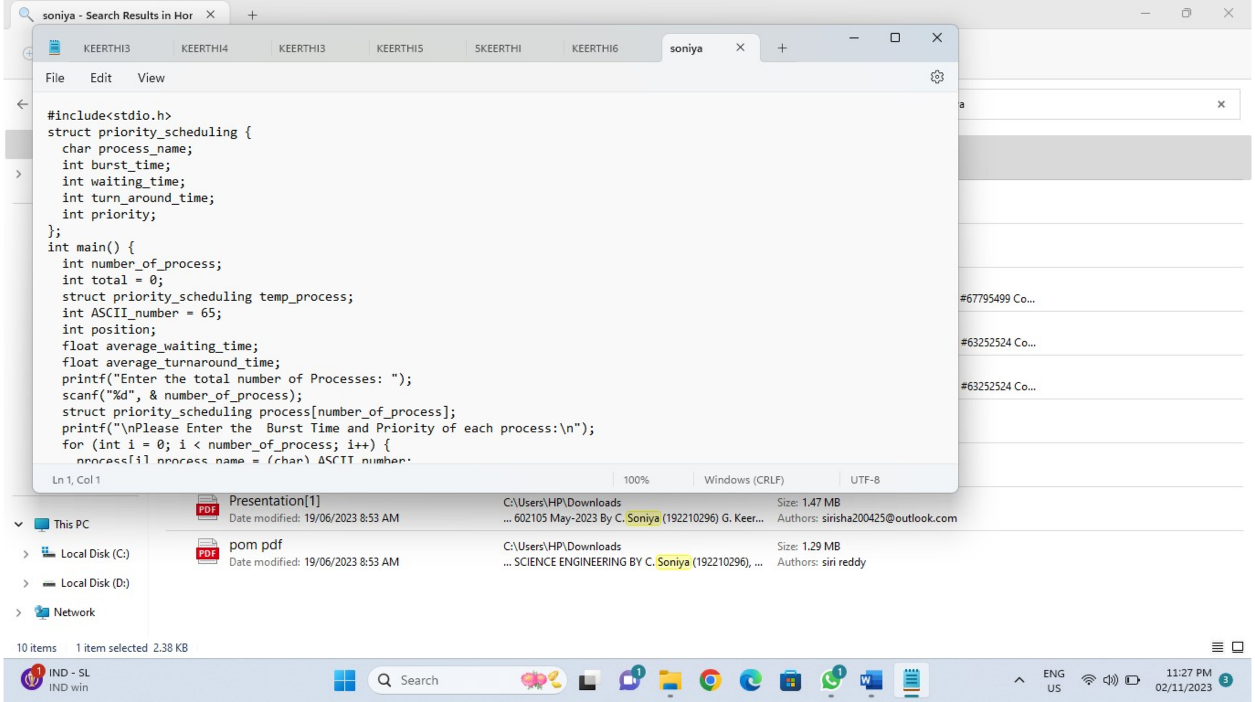
G . Keerthi Reddy- 192224165

Practical programs


C:\Users\HP\Documents\keerthi prog2.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
keerthi prog2.cpp
C:\Users\HP\Document
Line: 30 Col: 19 Sel: 0 Lines: 34 Length: 690 Insert Done parsing in 0.078 seconds
IND - SL
IND win
Search
ENG
US
11:26 PM
02/11/2023

```c
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

```
Enter the filename to open for reading
keerthi5.c
Enter the filename to open for writing
soniya

Contents copied to soniya
--------------------------------
Process exited after 9.026 seconds with return v
alue 0
Press any key to continue . . .
```



```c
#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;
```

Presentation[1] Date modified: 19/06/2023 8:53 AM C:\Users\HP\Downloads ... 602105 May-2023 By C. Soniya (192210296) G. Keer... Size: 1.47 MB Authors: sirisha200425@outlook.com

pom pdf Date modified: 19/06/2023 8:53 AM C:\Users\HP\Downloads ... SCIENCE ENGINEERING BY C. Soniya (192210296), ... Size: 1.29 MB Authors: siri reddy

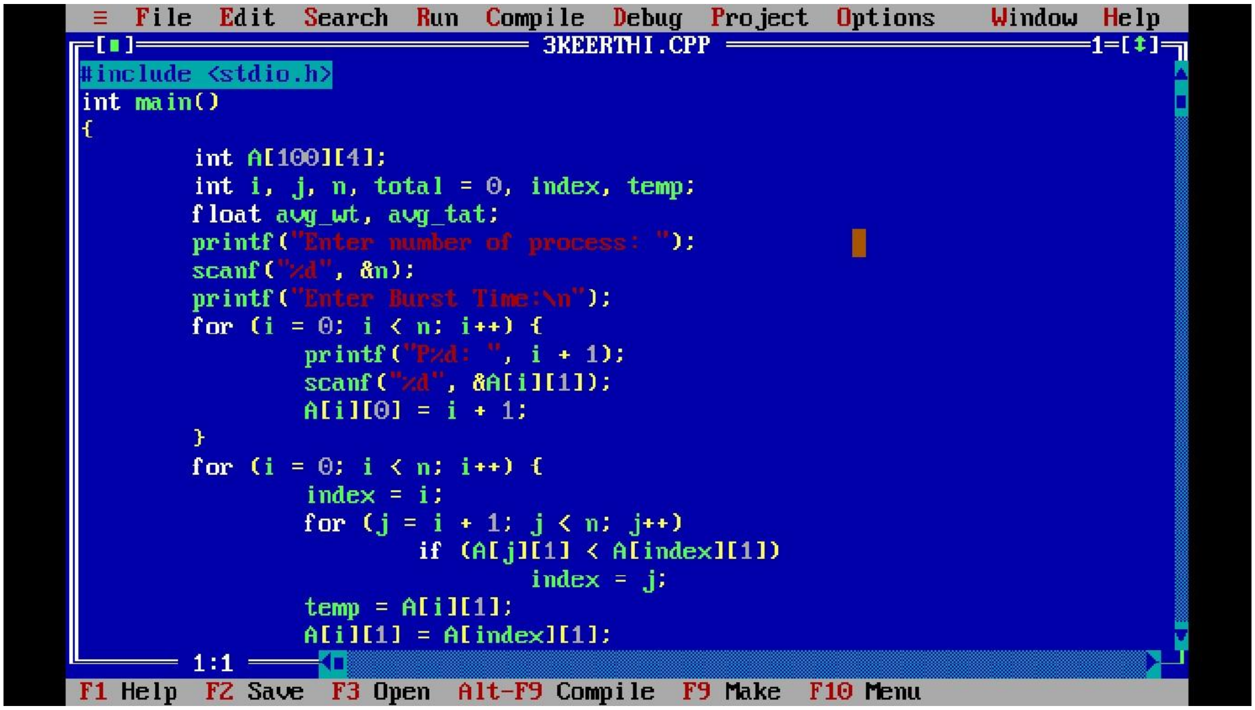
G . Keerthi Reddy- 192224165

Practical programs

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.

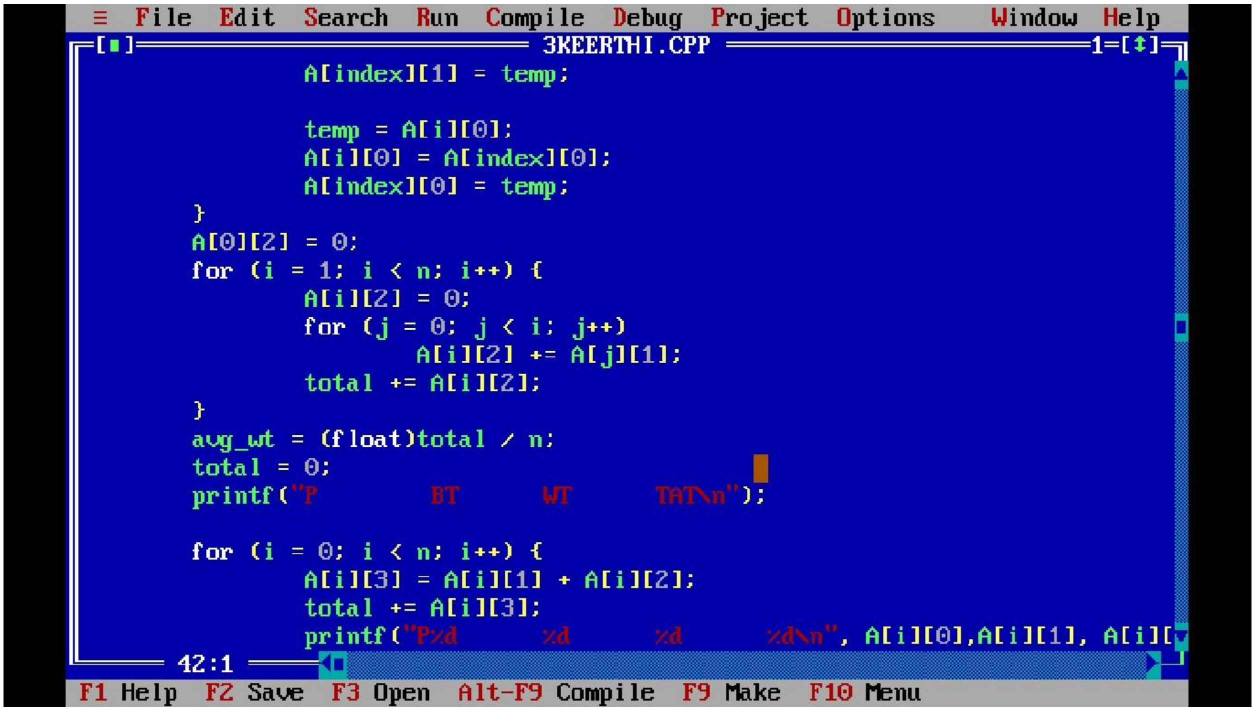
b. Assume that no process waits on I/O devices



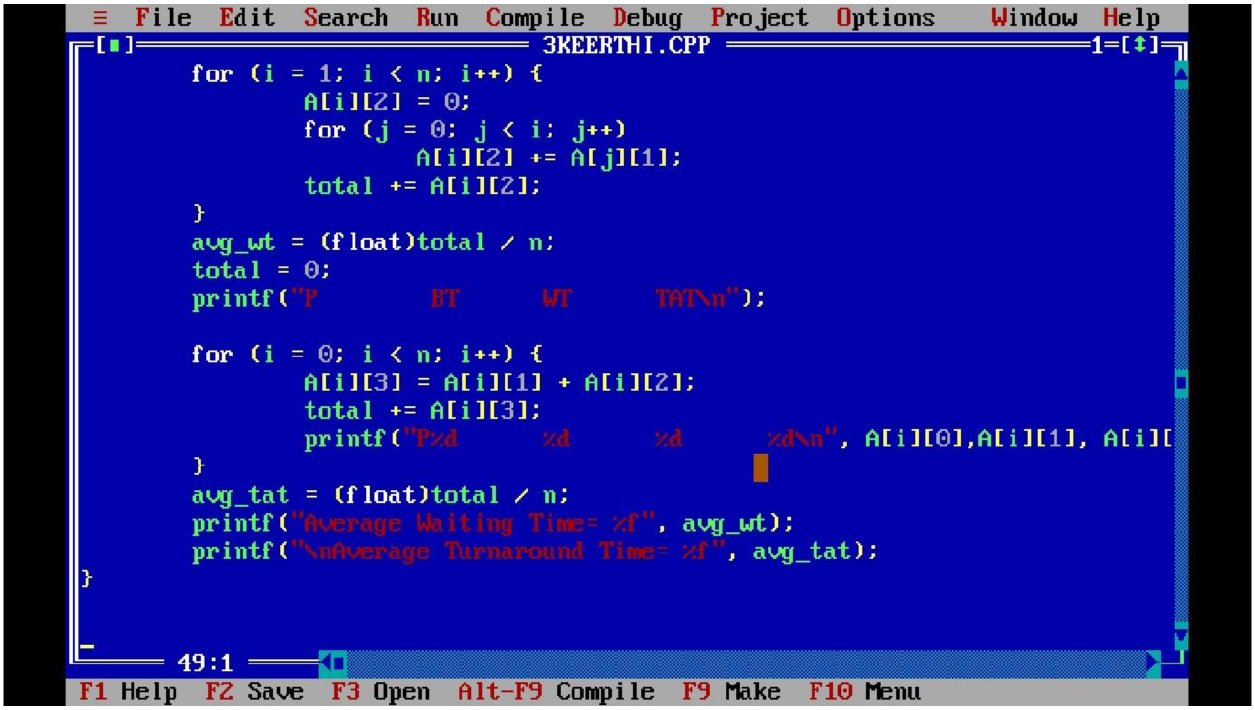
≡ File Edit Search Run Compile Debug Project Options Window Help
3KEERTHI.CPP
1=[‡]
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include <stdio.h>
int main()
{
        int A[100][4];
        int i, j, n, total = 0, index, temp;
        float avg_wt, avg_tat;
        printf("Enter number of process: ");
        scanf("%d", &n);
        printf("Enter Burst Time:\n");
        for (i = 0; i < n; i++) {
                printf("P%d: ", i + 1);
                scanf("%d", &A[i][1]);
                A[i][0] = i + 1;
        }
        for (i = 0; i < n; i++) {
                index = i;
                for (j = i + 1; j < n; j++)
                        if (A[j][1] < A[index][1])
                                index = j;
                temp = A[i][1];
                A[i][1] = A[index][1];
```

Practical programs

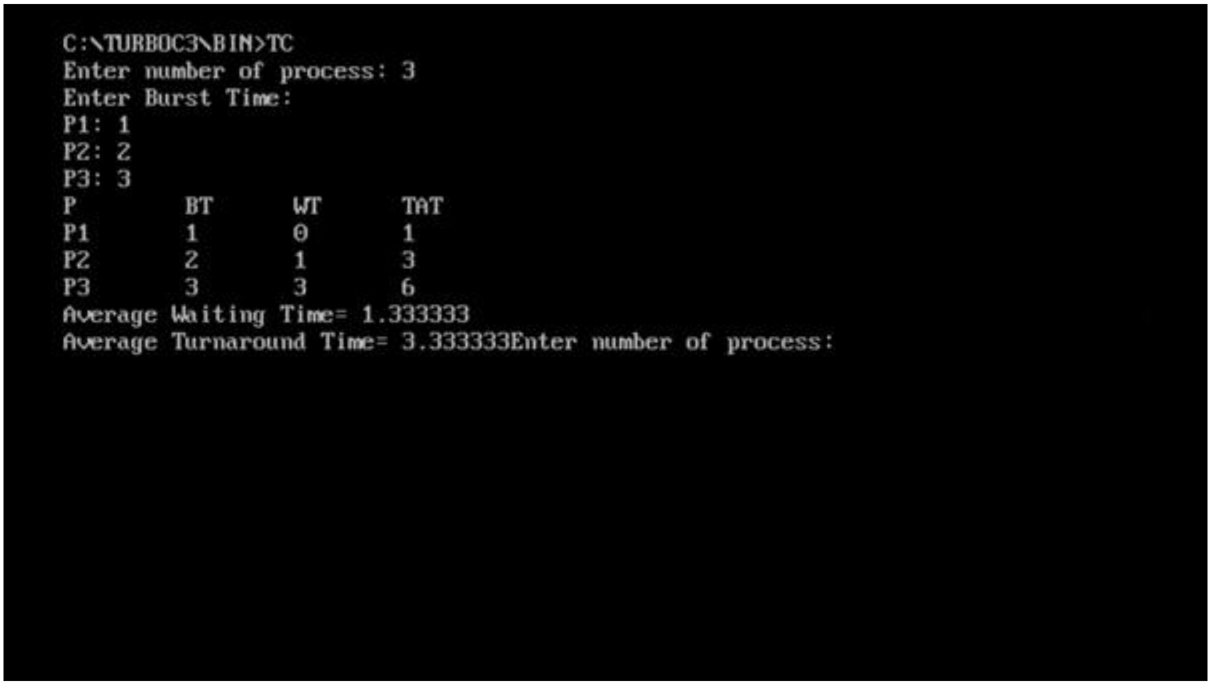

≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
3KEERTHI.CPP
1=[↕]
42:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
                A[index][1] = temp;

                temp = A[i][0];
                A[i][0] = A[index][0];
                A[index][0] = temp;
        }
        A[0][2] = 0;
        for (i = 1; i < n; i++) {
                A[i][2] = 0;
                for (j = 0; j < i; j++)
                        A[i][2] += A[j][1];
                total += A[i][2];
        }
        avg_wt = (float)total / n;
        total = 0;
        printf("P        BT      WT      TAT\n");

        for (i = 0; i < n; i++) {
                A[i][3] = A[i][1] + A[i][2];
                total += A[i][3];
                printf("P%d      %d      %d      %d\n", A[i][0],A[i][1], A[i][
```


≡ File Edit Search Run Compile Debug Project Options Window Help
[■]
3KEERTHI.CPP
1=[↕]
49:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

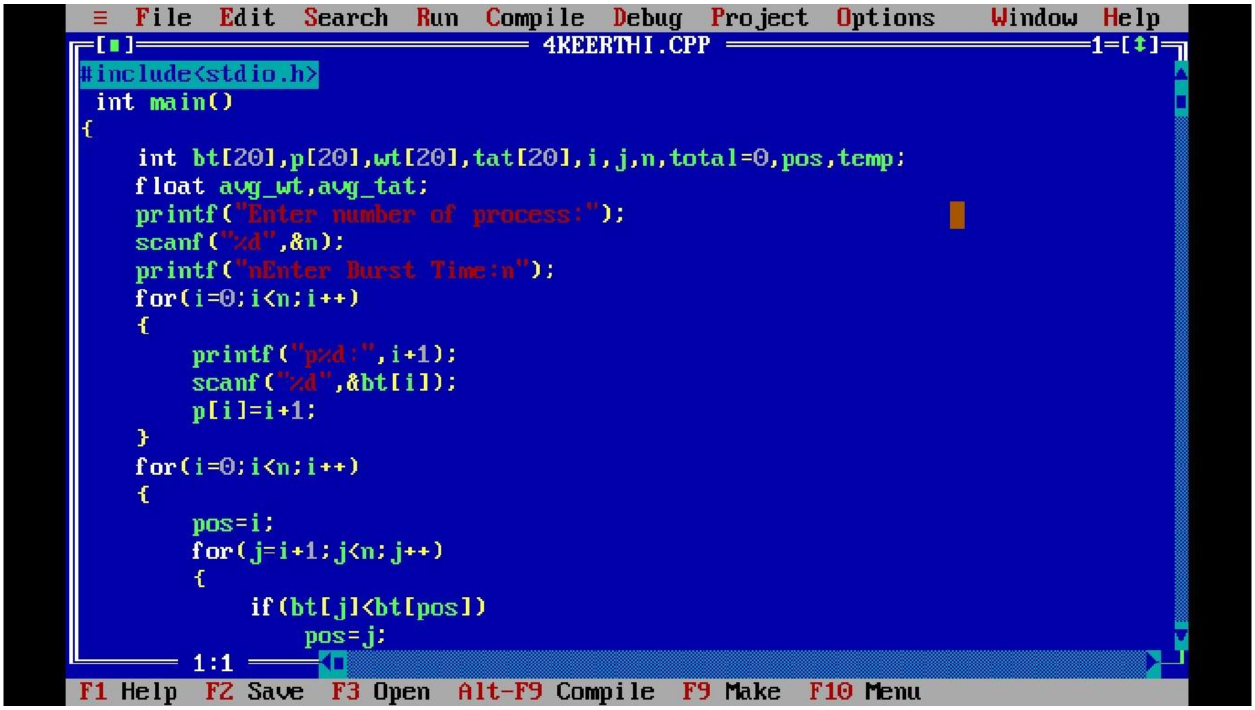
```
        for (i = 1; i < n; i++) {
                A[i][2] = 0;
                for (j = 0; j < i; j++)
                        A[i][2] += A[j][1];
                total += A[i][2];
        }
        avg_wt = (float)total / n;
        total = 0;
        printf("P        BT      WT      TAT\n");

        for (i = 0; i < n; i++) {
                A[i][3] = A[i][1] + A[i][2];
                total += A[i][3];
                printf("P%d      %d      %d      %d\n", A[i][0],A[i][1], A[i][
        }
        avg_tat = (float)total / n;
        printf("Average Waiting Time= %f", avg_wt);
        printf("\nAverage Turnaround Time= %f", avg_tat);
}
```

Practical programs



```
C:\TURBOC3\BIN>TC
Enter number of process: 3
Enter Burst Time:
P1: 1
P2: 2
P3: 3
P         BT       WT       TAT
P1        1        0        1
P2        2        1        3
P3        3        3        6
Average Waiting Time= 1.333333
Average Turnaround Time= 3.333333Enter number of process:
```

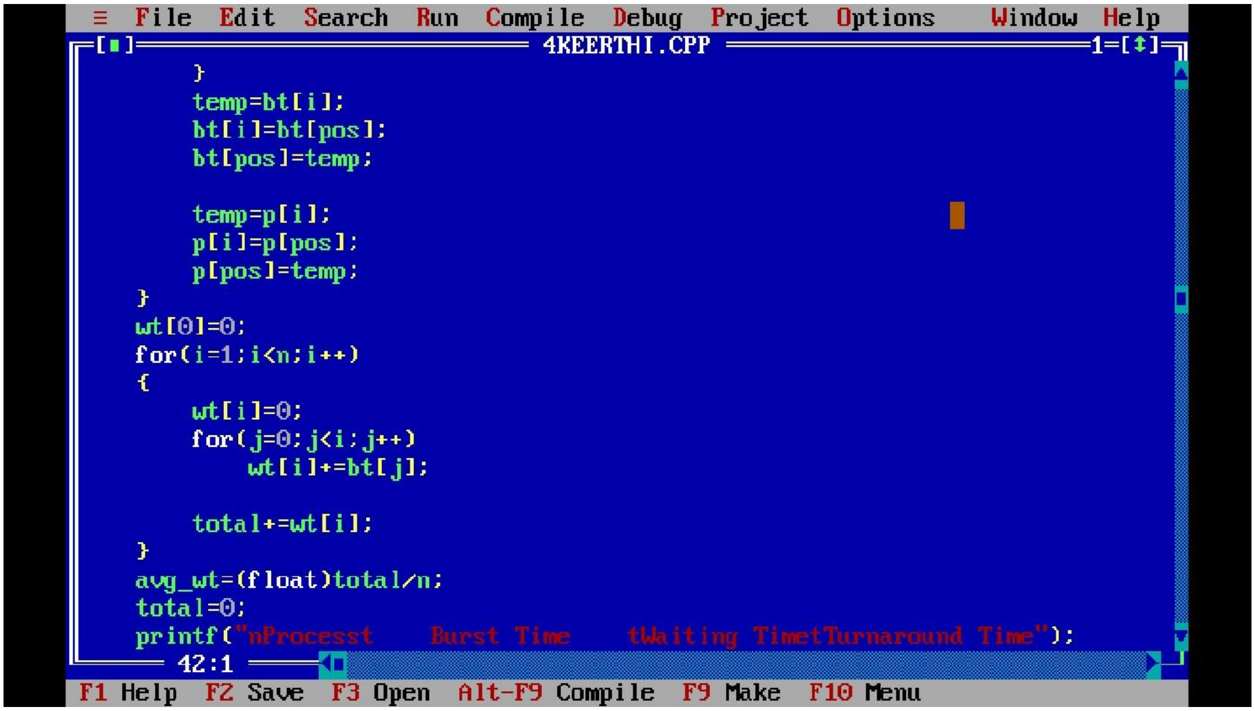
4. Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.



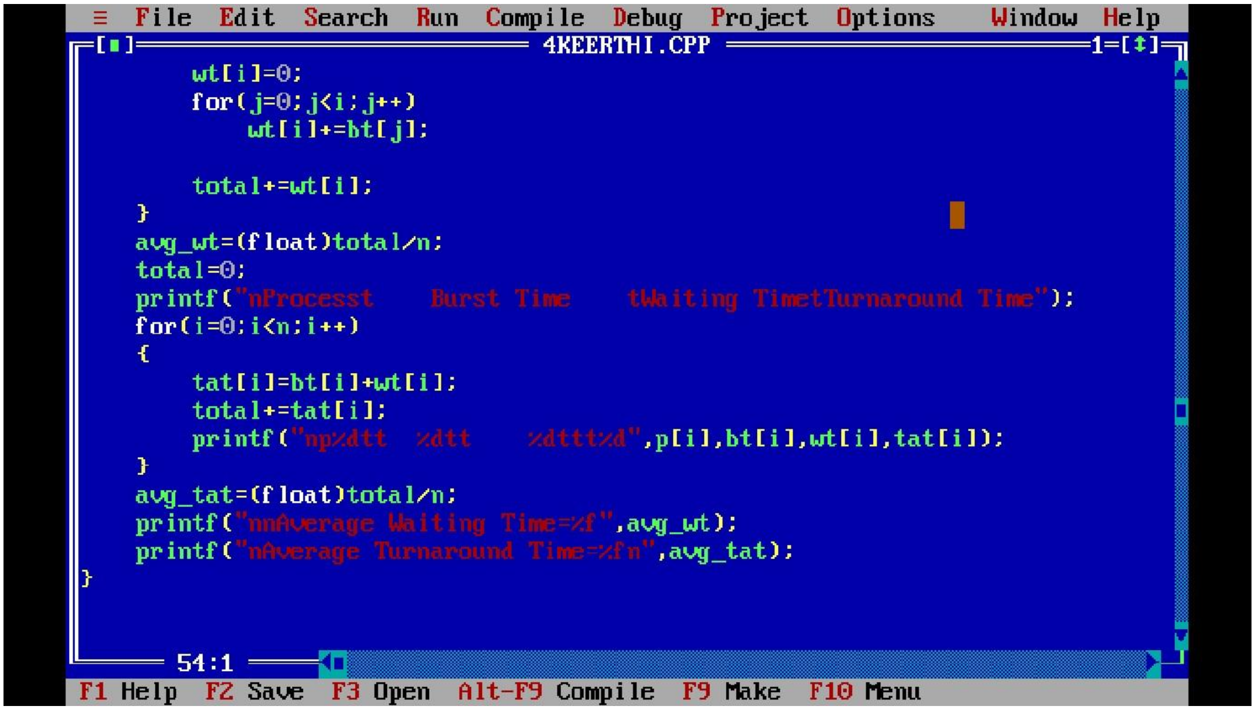
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] 4KEERTHI.CPP 1=[↕]
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#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;
```

Practical programs

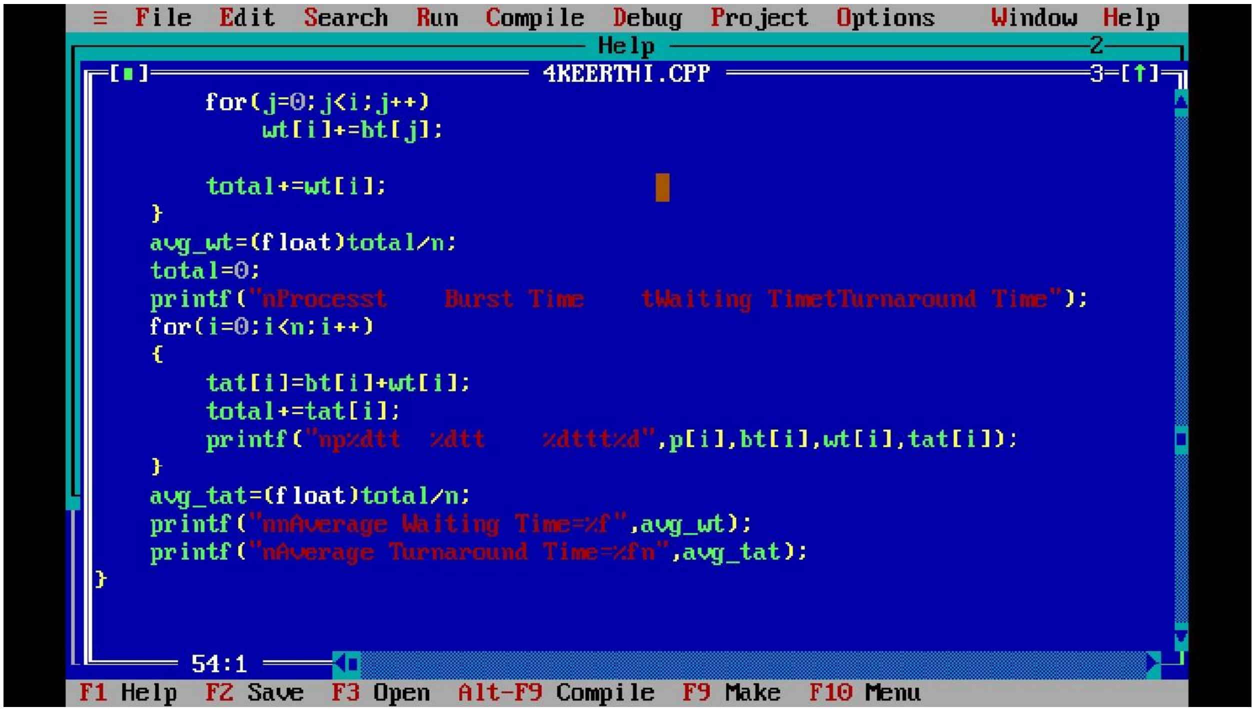

≡ File Edit Search Run Compile Debug Project Options Window Help
[■] 4KEERTHI.CPP 1=[↕]
42:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
        }
        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("nProcesst    Burst Time    tWaiting TimetTurnaround Time");
```

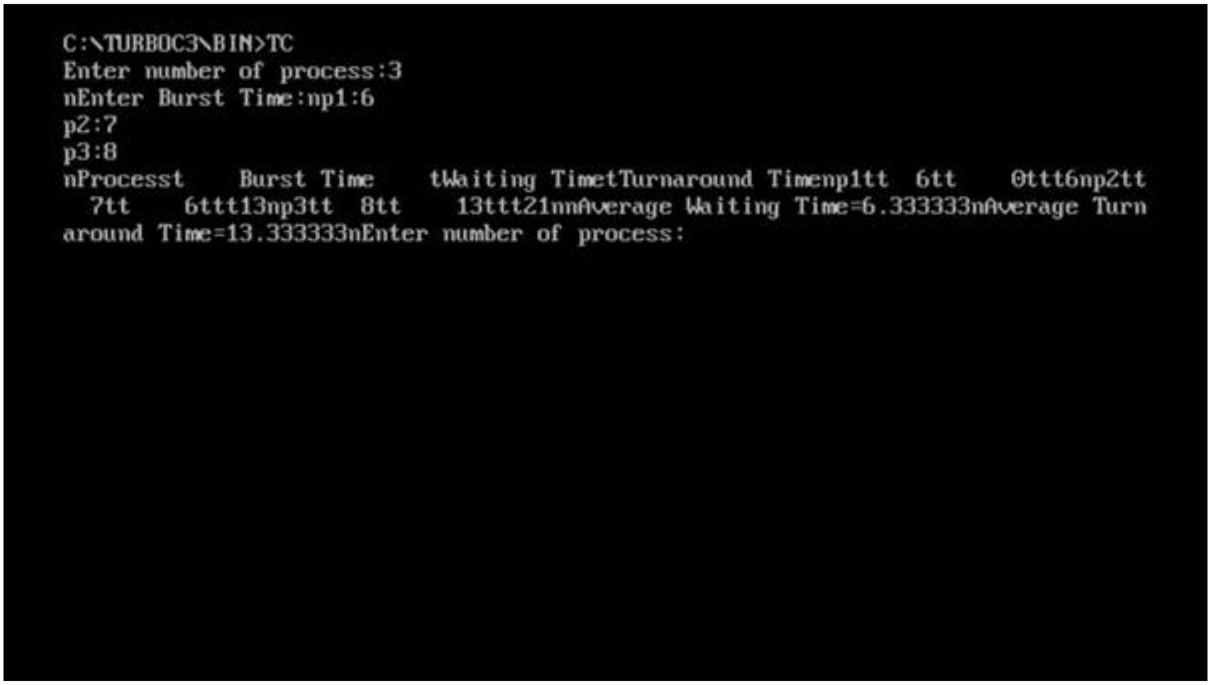

≡ File Edit Search Run Compile Debug Project Options Window Help
[■] 4KEERTHI.CPP 1=[↕]
54:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }
    avg_wt=(float)total/n;
    total=0;
    printf("nProcesst    Burst Time    tWaiting TimetTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("np%dtt  %dtt    %dttt%d",p[i],bt[i],wt[i],tat[i]);
    }
    avg_tat=(float)total/n;
    printf("nnAverage Waiting Time=%f",avg_wt);
    printf("nAverage Turnaround Time=%fn",avg_tat);
}
```

Practical programs


≡ File Edit Search Run Compile Debug Project Options Window Help
Help
2
[■]
4KEERTHI.CPP
3=[↑]
54:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

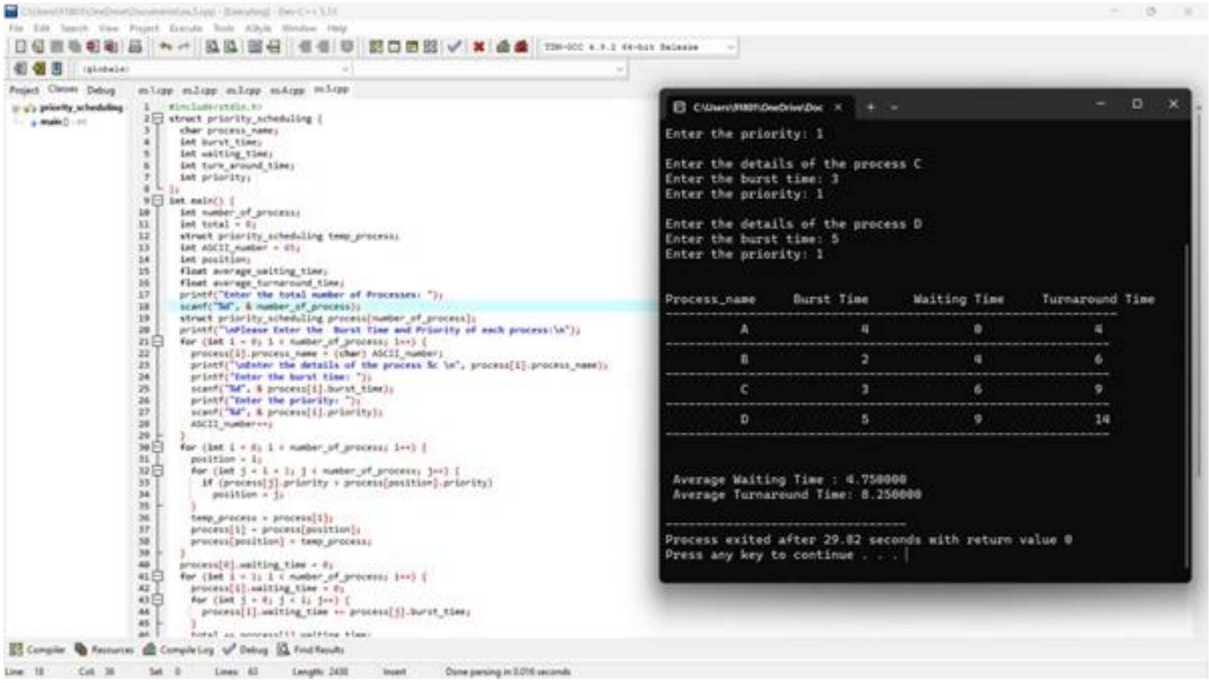
```
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }
    avg_wt=(float)total/n;
    total=0;
    printf("nProcesst     Burst Time     tWaiting TimetTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("np%dtt  %dtt    %dttt%d",p[i],bt[i],wt[i],tat[i]);
    }
    avg_tat=(float)total/n;
    printf("nnAverage Waiting Time=%f",avg_wt);
    printf("nAverage Turnaround Time=%fn",avg_tat);
}
```



```
C:\TURBOC3\BIN>TC
Enter number of process:3
nEnter Burst Time:np1:6
p2:7
p3:8
nProcesst    Burst Time    tWaiting TimetTurnaround Timenp1tt  6tt    0ttt6np2tt
  7tt    6ttt13np3tt  8tt    13ttt21nnAverage Waiting Time=6.333333nAverage Turn
around Time=13.333333nEnter number of process:
```

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

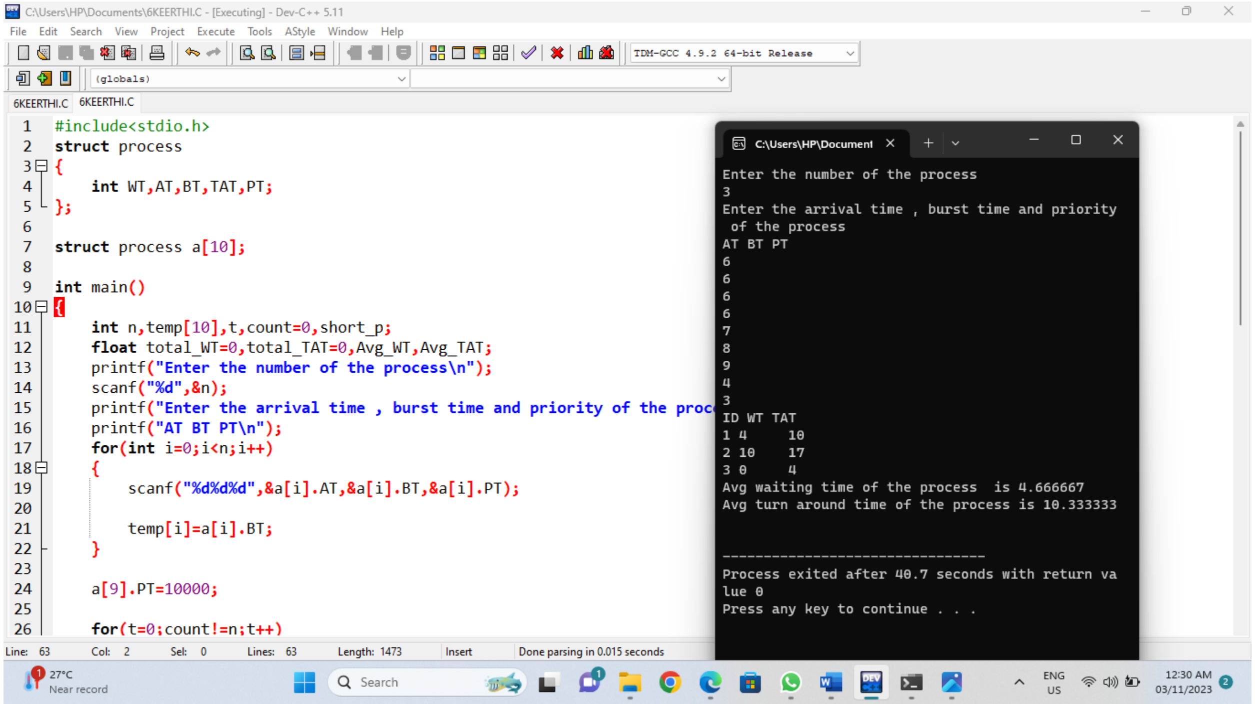
Practical programs



```
#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;
```

```
Enter the priority: 1

Enter the details of the process C
Enter the burst time: 3
Enter the priority: 1

Enter the details of the process D
Enter the burst time: 5
Enter the priority: 1

Process_name     Burst Time      Waiting Time     Turnaround Time
----------------------------------------------------------------
      A              4                0                4
----------------------------------------------------------------
      B              2                4                6
----------------------------------------------------------------
      C              3                6                9
----------------------------------------------------------------
      D              5                9               14
----------------------------------------------------------------


 Average Waiting Time : 4.750000
 Average Turnaround Time: 8.250000

--------------------------------
Process exited after 29.82 seconds with return value 0
Press any key to continue . . .
```

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



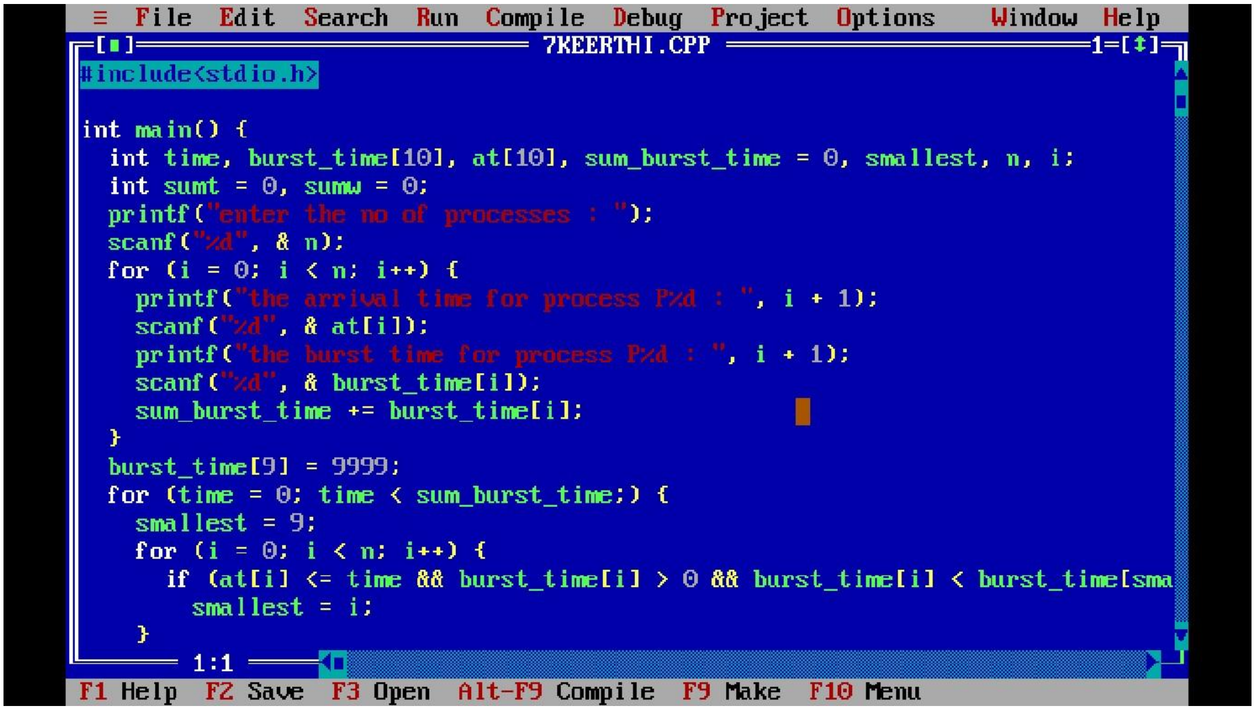
```
#include<stdio.h>
struct process
{
	int WT,AT,BT,TAT,PT;
};

struct process a[10];

int main()
{
	int n,temp[10],t,count=0,short_p;
	float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
	printf("Enter the number of the process\n");
	scanf("%d",&n);
	printf("Enter the arrival time , burst time and priority of the proc
	printf("AT BT PT\n");
	for(int i=0;i<n;i++)
	{
		scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

		temp[i]=a[i].BT;
	}

	a[9].PT=10000;

	for(t=0;count!=n;t++)
```

```
Enter the number of the process
3
Enter the arrival time , burst time and priority
 of the process
AT BT PT
6
6
6
6
7
8
9
4
3
ID WT TAT
1 4     10
2 10    17
3 0     4
Avg waiting time of the process  is 4.666667
Avg turn around time of the process is 10.333333


--------------------------------
Process exited after 40.7 seconds with return va
lue 0
Press any key to continue . . .
```

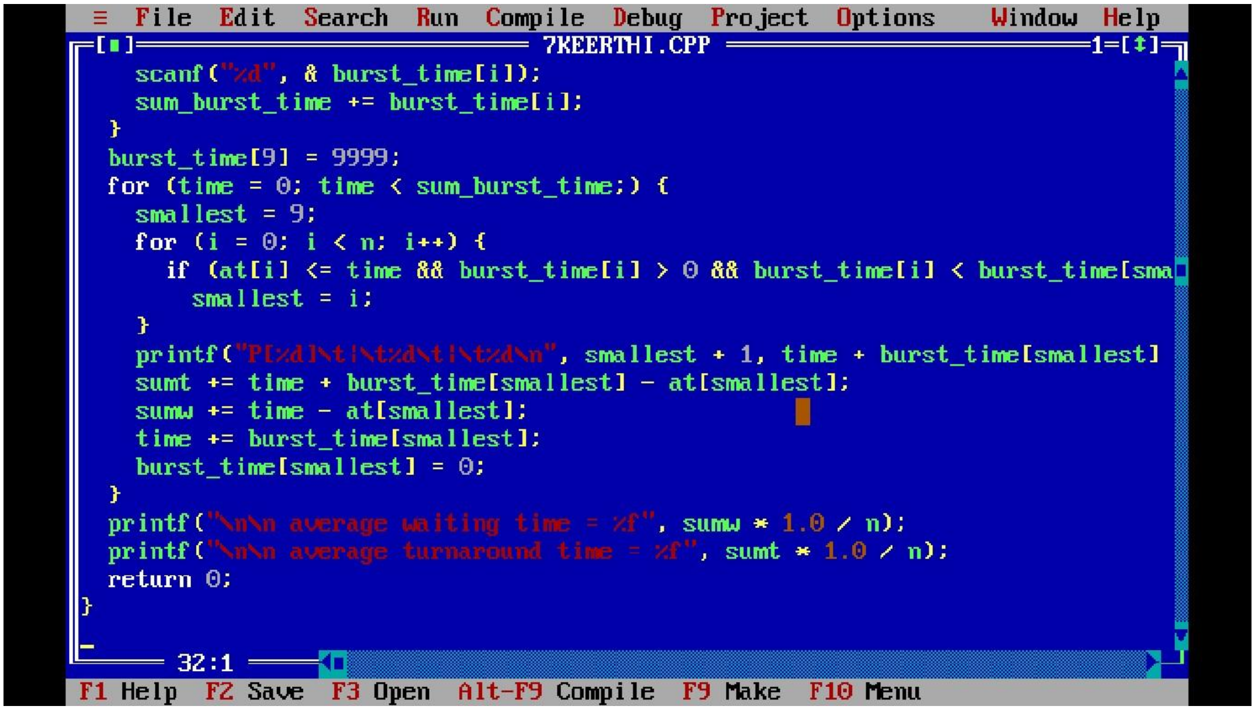
G . Keerthi Reddy- 192224165

Practical programs

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

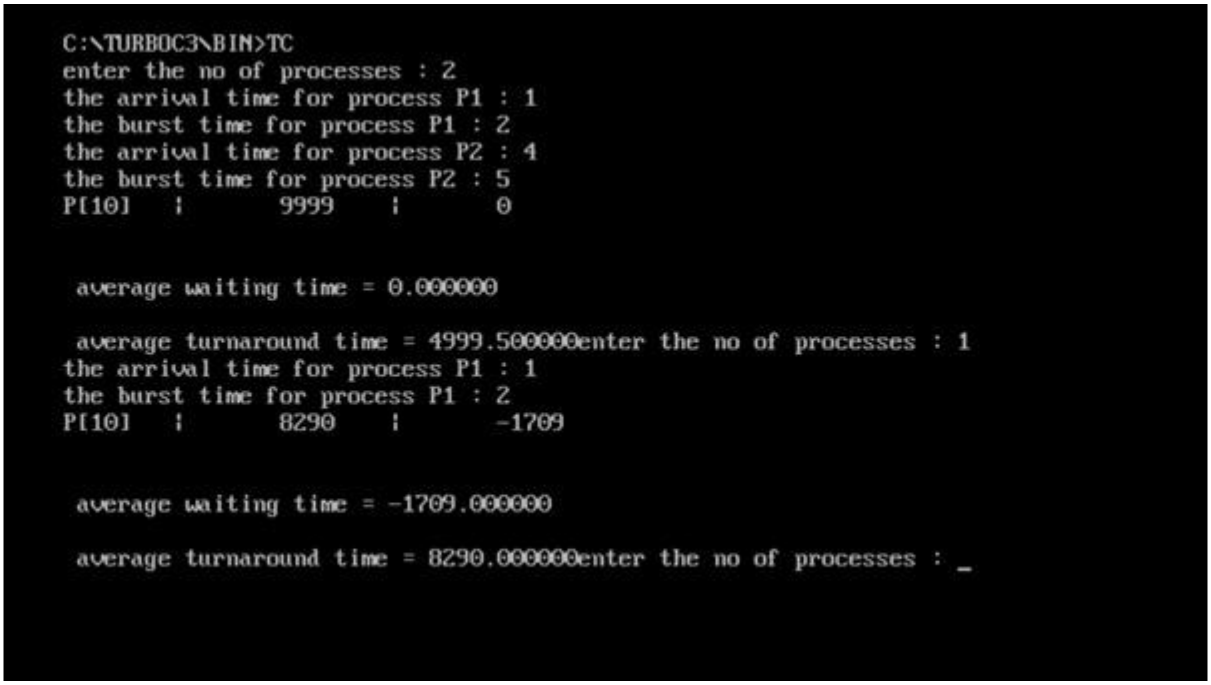

≡ File Edit Search Run Compile Debug Project Options Window Help
7KEERTHI.CPP
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#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[sma
        smallest = i;
    }
```


≡ File Edit Search Run Compile Debug Project Options Window Help
7KEERTHI.CPP
32:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

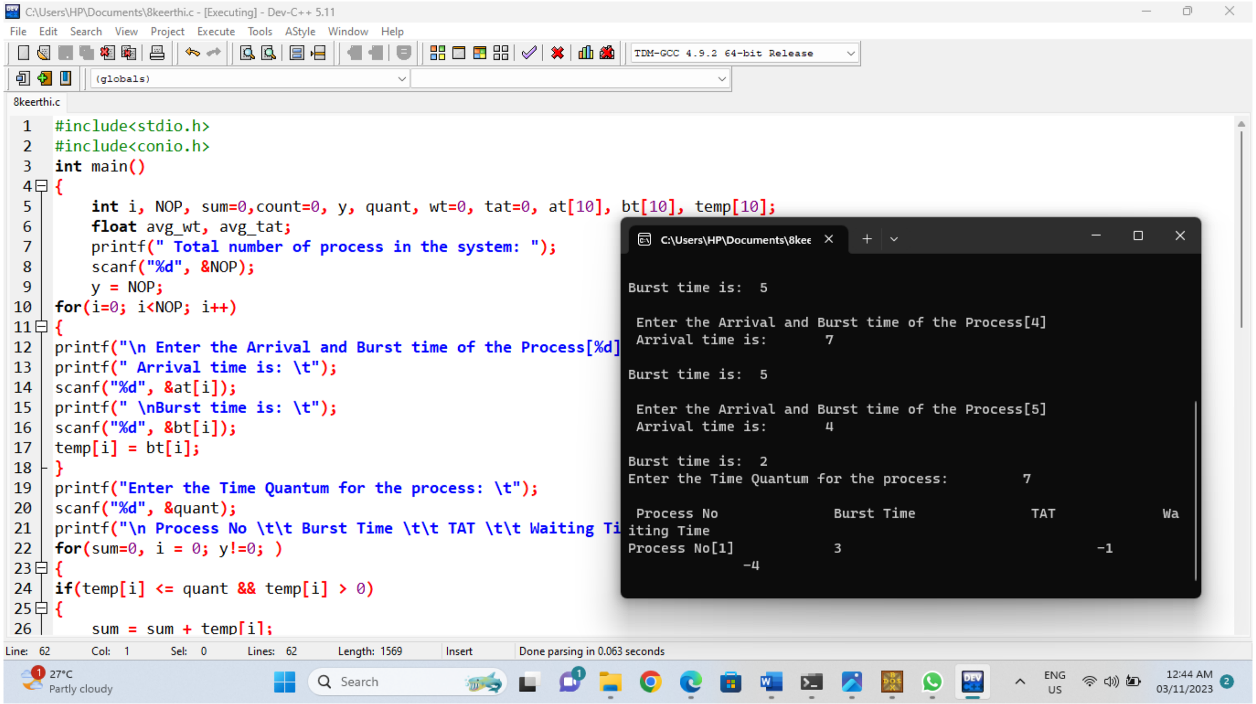
```
    scanf("%d", & burst_time[i]);
    sum_burst_time += burst_time[i];
  }
  burst_time[9] = 9999;
  for (time = 0; time < sum_burst_time;) {
    smallest = 9;
    for (i = 0; i < n; i++) {
      if (at[i] <= time && burst_time[i] > 0 && burst_time[i] < burst_time[sma
        smallest = i;
    }
    printf("P[%d]\t|\t%d\t|\t%d\n", smallest + 1, time + burst_time[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;
}
```

Practical programs



```
C:\TURBOC3\BIN>TC
enter the no of processes : 2
the arrival time for process P1 : 1
the burst time for process P1 : 2
the arrival time for process P2 : 4
the burst time for process P2 : 5
P[10]   |       9999    |       0


 average waiting time = 0.000000

 average turnaround time = 4999.500000enter the no of processes : 1
the arrival time for process P1 : 1
the burst time for process P1 : 2
P[10]   |       8290    |       -1709


 average waiting time = -1709.000000

 average turnaround time = 8290.000000enter the no of processes : _
```

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



C:\Users\HP\Documents\8keerthi.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
8keerthi.c
Line: 62 Col: 1 Sel: 0 Lines: 62 Length: 1569 Insert Done parsing in 0.063 seconds
27°C Partly cloudy
Search
ENG US
12:44 AM 03/11/2023

```c
#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]
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 Ti
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0)
{
    sum = sum + temp[i];
```

```
Burst time is:  5

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       7

Burst time is:  5

 Enter the Arrival and Burst time of the Process[5]
 Arrival time is:       4

Burst time is:  2
Enter the Time Quantum for the process:        7

 Process No             Burst Time              TAT             Wa
iting Time
Process No[1]           3                               -1
              -4
```

G . Keerthi Reddy- 192224165

Practical programs

G . Keerthi Reddy- 192224165