Name: VENNELA G

Register No: 20BDS0146

Lab Course Name: OPERATING SYSTEMS

Lab Slot: L21+L22

Lab Assessment Title: PROCESSES & THREADS

Question 1

Code the following scheduling algorithms in C and print the performance parameters for an arbitrary set of inputs (Process IDs, Burst Times, Arrival Times, Priorities). Use of appropriate data structures will be appreciable.

a. FCFS

b. SJF

c. SRTF

d. PRIORITY

e. ROUND ROBIN

Answer:

a.

SOURCE CODE:

```
#include<stdio.h>

#include<string.h>


int main()

{


   int
arrival_t[10],burst_t[10],starting_t[10],finish_t[10],turnaround_t[10],waiting_t
[10];

   int i,j,n,temp;
```

```
int totalwaiting_t=0,totalturnaround_t=0;
char procname[10][10],timing[10];

printf("Enter the Number of Processes:");
scanf("%d",&n);
for(i=0; i<n; i++)
{
   printf("Enter the Processname along with ID, Arrival Time & Burst Time:");
   scanf("%s%d%d",procname[i],&arrival_t[i],&burst_t[i]);
}
for(i=0; i<n; i++)
{
   for(j=0; j<n; j++)
   {
      if(arrival_t[i]<arrival_t[j])
      {
         temp=arrival_t[i];
         arrival_t[i]=arrival_t[j];
         arrival_t[j]=temp;
         temp=burst_t[i];
         burst_t[i]=burst_t[j];
         burst_t[j]=temp;
         strcpy(timing,procname[i]);
         strcpy(procname[i],procname[j]);
         strcpy(procname[j],timing);
      }
```

```
        }
    }
    for(i=0; i<n; i++)
    {
        if(i==0)
            starting_t[i]=arrival_t[i];
        else
            starting_t[i]=finish_t[i-1];
        waiting_t[i]=starting_t[i]-arrival_t[i];
        finish_t[i]=starting_t[i]+burst_t[i];
        turnaround_t[i]=finish_t[i]-arrival_t[i];
    }
    printf("\nProcess\tArrivaltime\t Bursttime\t WaitTime\t Turnaround Time");
    for(i=0; i<n; i++)
    {

printf("\n%s\t%3d\t%3d\t%3d\t%3d\t%6d\t%6d",procname[i],arrival_t[i],burs
t_t[i],waiting_t[i],starting_t[i],turnaround_t[i],finish_t[i]);
        totalwaiting_t+=waiting_t[i];
        totalturnaround_t+=turnaround_t[i];
       }
    printf("\nAverage Waiting time:%f",(float)totalwaiting_t/n);
    printf("\nAverage Turn Around Time:%f",(float)totalturnaround_t/n);
    return 0;
}
```

OUTPUT:

```
vennela@vennela-VirtualBox:~/CSE2005$ gcc ex1a.c
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
Enter the Number of Processes:4
Enter the Processname along with ID, Arrival Time & Burst Time:process1 2
4
Enter the Processname along with ID, Arrival Time & Burst Time:process3 5 10
Enter the Processname along with ID, Arrival Time & Burst Time:process2 6 3
Enter the Processname along with ID, Arrival Time & Burst Time:process4 3 8

Process Arrivaltime      Bursttime       WaitTime        Turnaround Time
process1          2       4       0       2          4       6
process4          3       8       3       6         11      14
process3          5      10       9      14         19      24
process2          6       3      18      24         21      27
Average Waiting time:7.500000
Average Turn Around Time:13.750000vennela@vennela-VirtualBox:~/CSE2005$
```

b.

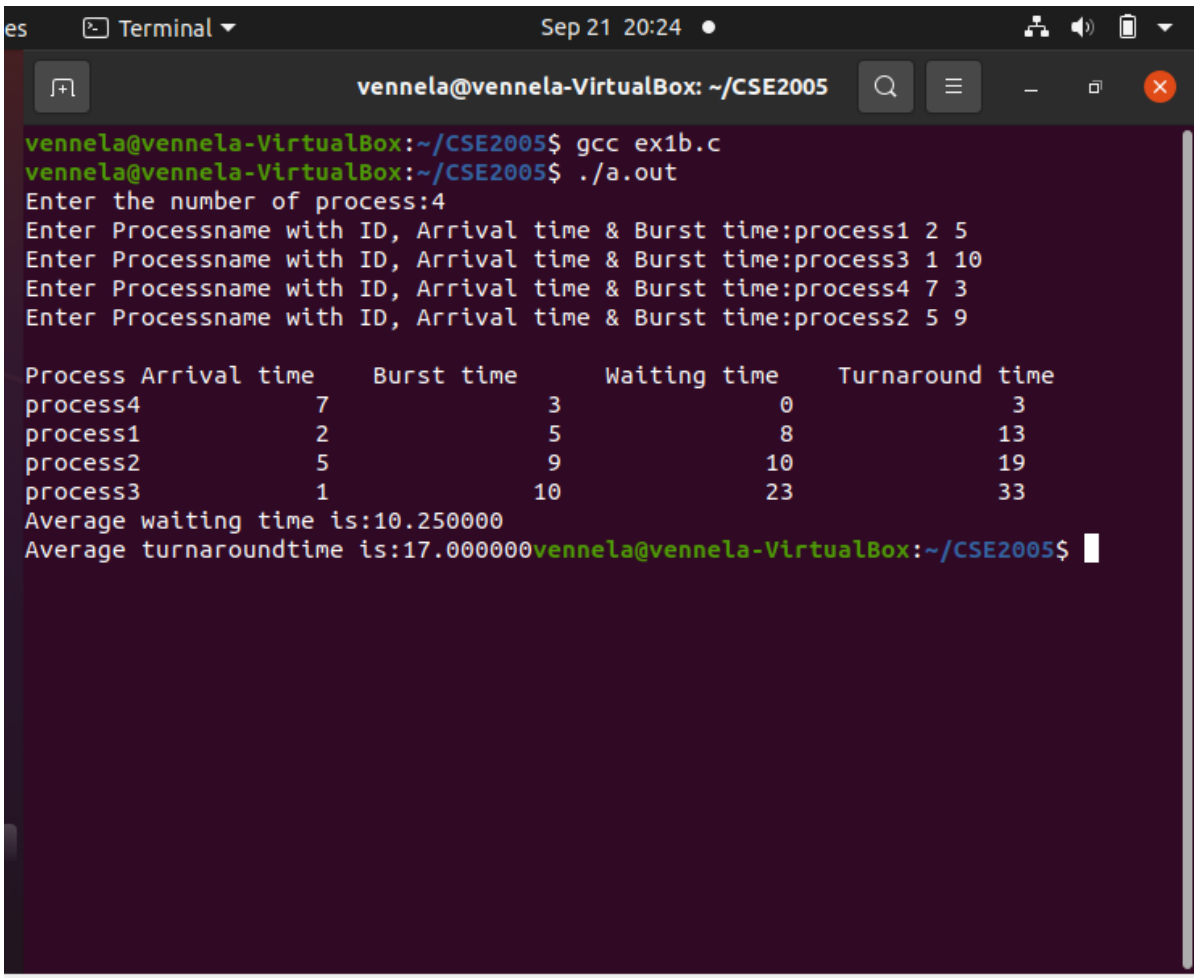
SOURCE CODE:

```
#include<stdio.h>

#include<string.h>

void main()

{
```

```
    int
burst_t[20],arrival_t[10],starting_t[10],finish_t[10],waiting_t[10],turnaround_t
[10];
    int n,i,j,temp;
    int totalwaiting_t=0,totalturnaround_t=0;
    float avgwaiting_t,avgturnaround_t;
    char procname[10][10],timing[10];

    printf("Enter the number of process:");
    scanf("%d",&n);
    for(i=0; i<n; i++)
    {
        printf("Enter Processname with ID, Arrival time & Burst time:");

        scanf("%s%d%d",procname[i],&arrival_t[i],&burst_t[i]);
    }
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
        {
            if(burst_t[i]<burst_t[j])
            {
                temp=arrival_t[i];
                arrival_t[i]=arrival_t[j];
                arrival_t[j]=temp;
                temp=burst_t[i];
                burst_t[i]=burst_t[j];
                burst_t[j]=temp;
```

```
            strcpy(timing,procname[i]);
            strcpy(procname[i],procname[j]);
            strcpy(procname[j],timing);
        }
    }
  for(i=0; i<n; i++)
  {
    if(i==0)
        starting_t[i]=arrival_t[i];
    else
        starting_t[i]=finish_t[i-1];
    waiting_t[i]=starting_t[i]-arrival_t[i];
    finish_t[i]=starting_t[i]+burst_t[i];
    turnaround_t[i]=finish_t[i]-arrival_t[i];
    totalwaiting_t+=waiting_t[i];
    totalturnaround_t+=turnaround_t[i];
  }
  avgwaiting_t=(float)totalwaiting_t/n;
  avgturnaround_t=(float)totalturnaround_t/n;
  printf("\nProcess\tArrival time\tBurst time\tWaiting time\tTurnaround time");
  for(i=0; i<n; i++)

printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",procname[i],arrival_t[i],burst_t[i],waiting_t[i],turnaround_t[i]);
  printf("\nAverage waiting time is:%f",avgwaiting_t);
  printf("\nAverage turnaroundtime is:%f",avgturnaround_t);}
```

OUTPUT:



```
vennela@vennela-VirtualBox:~/CSE2005$ gcc ex1b.c
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
Enter the number of process:4
Enter Processname with ID, Arrival time & Burst time:process1 2 5
Enter Processname with ID, Arrival time & Burst time:process3 1 10
Enter Processname with ID, Arrival time & Burst time:process4 7 3
Enter Processname with ID, Arrival time & Burst time:process2 5 9

Process Arrival time    Burst time      Waiting time    Turnaround time
process4            7               3               0               3
process1            2               5               8               13
process2            5               9               10              19
process3            1               10              23              33
Average waiting time is:10.250000
Average turnaroundtime is:17.000000vennela@vennela-VirtualBox:~/CSE2005$
```

c.

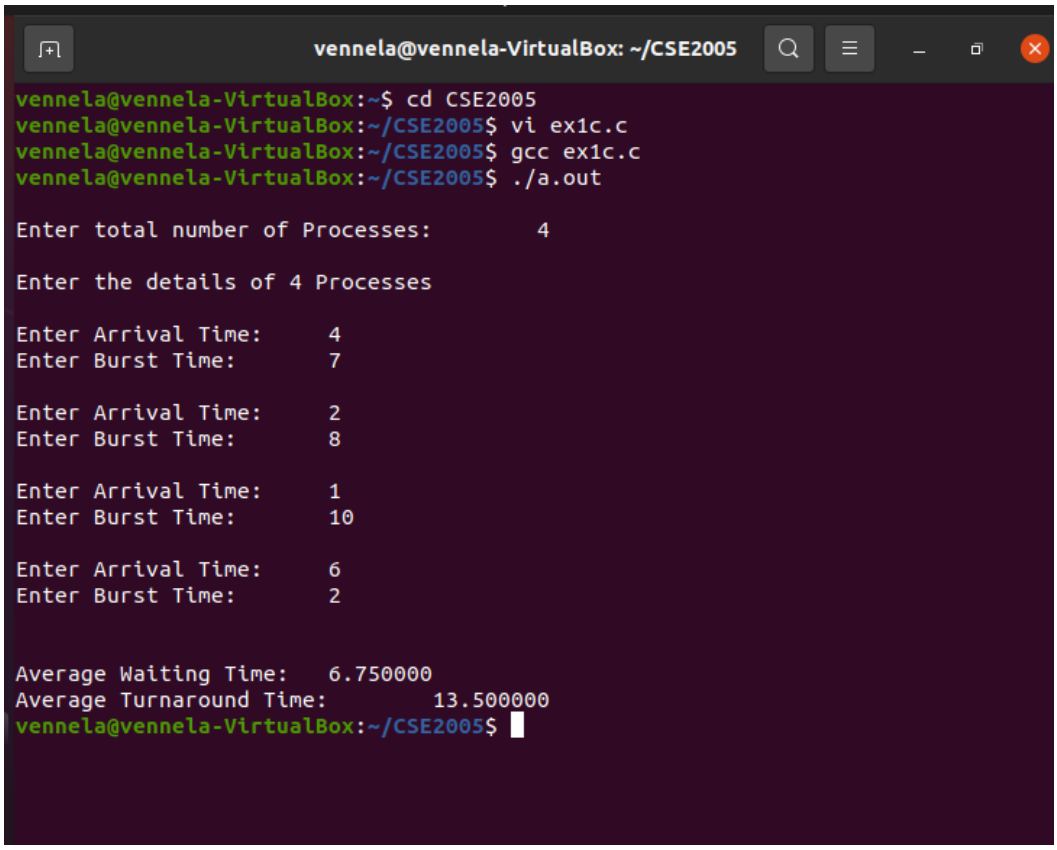
SOURCE CODE:

```
#include <stdio.h>

int main()

{

    int arrival_t[10], burst_t[10], temp[10];

    int i, small, flag = 0, timer, n;

    double waiting_t = 0, turnaround_t = 0, ending_t;
```

```
float avgwaiting_t, avgturnaround_t;
printf("\nEnter total number of Processes:\t");
scanf("%d", &n);
printf("\nEnter the details of %d Processes\n",n);
for(i = 0; i < n; i++)
{
    printf("\nEnter Arrival Time:\t");
    scanf("%d", &arrival_t[i]);
    printf("Enter Burst Time:\t");
    scanf("%d", &burst_t[i]);
    temp[i] = burst_t[i];
}
burst_t[9] = 9999;
for(timer = 0; flag != n; timer++)
{
    small = 9;
    for(i = 0; i < n; i++)
    {
        if(arrival_t[i] <= timer && burst_t[i] < burst_t[small] && burst_t[i] > 0)
        {
            small = i;
            }
    }
    burst_t[small]--;
    if(burst_t[small] == 0)
    {
```

```
            flag++;
            ending_t = timer + 1;
            waiting_t = waiting_t + ending_t - arrival_t[small] - temp[small];
            turnaround_t = turnaround_t + ending_t - arrival_t[small];
        }
    }
    avgwaiting_t = waiting_t / n;
    avgturnaround_t = turnaround_t / n;
    printf("\n\nAverage Waiting Time:\t%lf\n", avgwaiting_t);
    printf("Average Turnaround Time:\t%lf\n", avgturnaround_t);
    return 0;}
```

OUTPUT:



d.

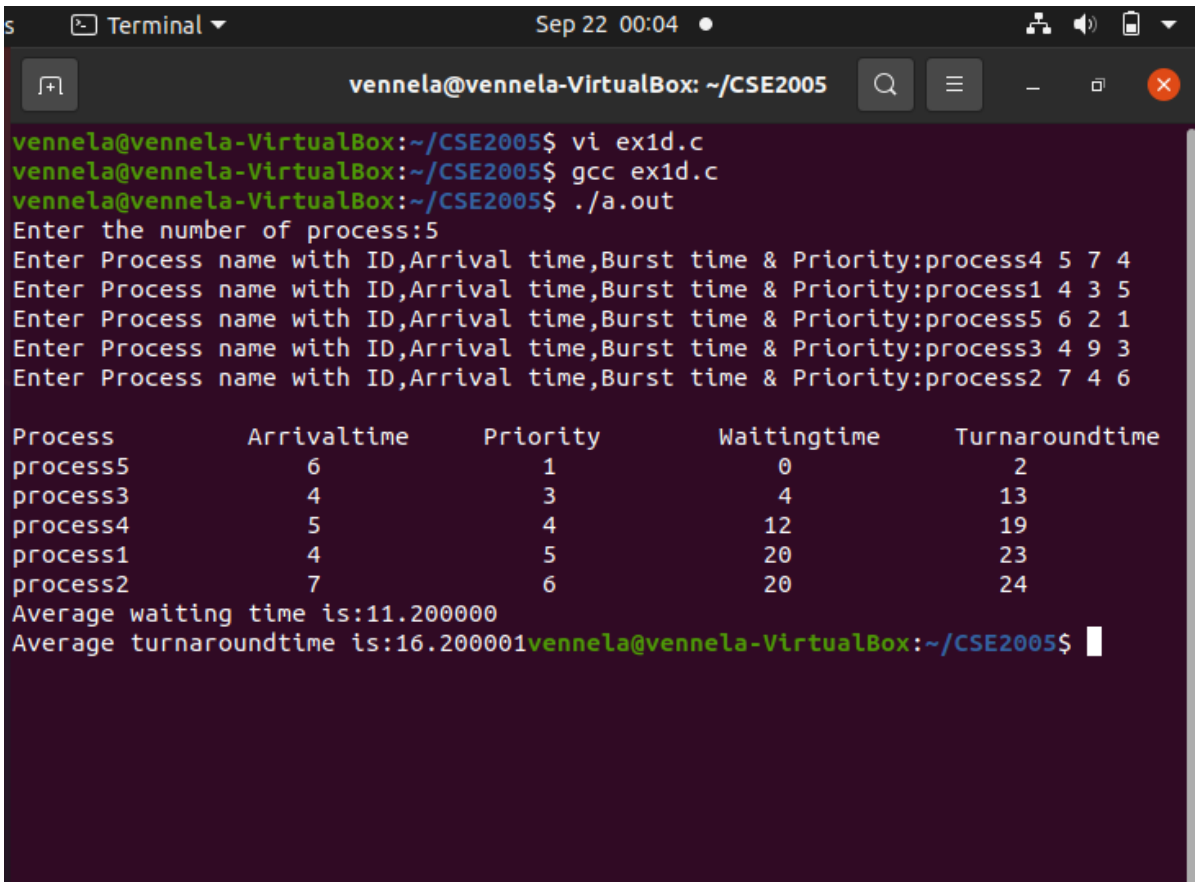
SOURCE CODE:

```
#include<stdio.h>

#include<string.h>

void main()

{

   int
burst_t[20],arrival_t[10],proc[10],starting_t[10],finish_t[10],waiting_t[10],turn
around_t[10];

   int n,i,j,temp;

   int totalwaiting_t=0,totalturnaround_t=0;

   float avgwaiting_t,avgturnaround_t;

   char procname[10][10],timing[10];


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

   scanf("%d",&n);

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

   {

      printf("Enter Process name with ID,Arrival time,Burst time & Priority:");


      scanf("%s%d%d%d",procname[i],&arrival_t[i],&burst_t[i],&proc[i]);

   }

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

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

      {

         if(proc[i]<proc[j])
```

```
        {
            temp=proc[i];
            proc[i]=proc[j];
            proc[j]=temp;
            temp=arrival_t[i];
            arrival_t[i]=arrival_t[j];
            arrival_t[j]=temp;
            temp=burst_t[i];
            burst_t[i]=burst_t[j];
            burst_t[j]=temp;
            strcpy(timing,procname[i]);
            strcpy(procname[i],procname[j]);
            strcpy(procname[j],timing);
        }
    }
for(i=0; i<n; i++)

{

    if(i==0)
    {
        starting_t[i]=arrival_t[i];
        waiting_t[i]=starting_t[i]-arrival_t[i];
        finish_t[i]=starting_t[i]+burst_t[i];
        turnaround_t[i]=finish_t[i]-arrival_t[i];
    }
```

```
        else
        {
            starting_t[i]=finish_t[i-1];
            waiting_t[i]=starting_t[i]-arrival_t[i];
            finish_t[i]=starting_t[i]+burst_t[i];
            turnaround_t[i]=finish_t[i]-arrival_t[i];
            }
        totalwaiting_t+=waiting_t[i];
        totalturnaround_t+=turnaround_t[i];
    }
    avgwaiting_t=(float)totalwaiting_t/n;
    avgturnaround_t=(float)totalturnaround_t/n;
    printf("\nProcess\tArrivaltime\tPriority\tWaitingtime\tTurnaroundtime");
    for(i=0; i<n; i++)
        printf("\n%s\t%5d
            \t\t%5d\t\t%5d\t\t%5d",procname[i],arrival_t[i],proc[i],waiting_t[i],turn
around_t[i]);
    printf("\nAverage waiting time is:%f",avgwaiting_t);
    printf("\nAverage turnaroundtime is:%f",avgturnaround_t);


}
```

OUTPUT:



```
vennela@vennela-VirtualBox:~/CSE2005$ vi ex1d.c
vennela@vennela-VirtualBox:~/CSE2005$ gcc ex1d.c
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
Enter the number of process:5
Enter Process name with ID,Arrival time,Burst time & Priority:process4 5 7 4
Enter Process name with ID,Arrival time,Burst time & Priority:process1 4 3 5
Enter Process name with ID,Arrival time,Burst time & Priority:process5 6 2 1
Enter Process name with ID,Arrival time,Burst time & Priority:process3 4 9 3
Enter Process name with ID,Arrival time,Burst time & Priority:process2 7 4 6

Process         Arrivaltime     Priority        Waitingtime     Turnaroundtime
process5            6               1               0               2
process3            4               3               4               13
process4            5               4               12              19
process1            4               5               20              23
process2            7               6               20              24
Average waiting time is:11.200000
Average turnaroundtime is:16.200001vennela@vennela-VirtualBox:~/CSE2005$
```

e.

SOURCE CODE:

```
#include<stdio.h>

int main()

{

    int i, n, total = 0, n1, flag = 0, timer;

    int waiting_t = 0, turnaround_t = 0, arrival_t[10], burst_t[10], temp[10];

    float avgwaiting_t, avgturnaround_t;

    printf("\nEnter the total number of Processes:\t");
```

```
scanf("%d", &n);
n1 = n;
for(i = 0; i < n; i++)
{
    printf("\nEnter the details of Processes\n");
    printf("Arrival Time:\t");
    scanf("%d", &arrival_t[i]);
    printf("Burst Time:\t");
    scanf("%d", &burst_t[i]);
    temp[i] = burst_t[i];
}
printf("\nEnter Time limit:\t");
scanf("%d", &timer);
printf("\nProcessID\tBursttime\tTurnaroundTime\t WaitingTime\n");
for(total = 0, i = 0; n1 != 0;)
{
    if(temp[i] <= timer && temp[i] > 0)
    {
        total = total + temp[i];
        temp[i] = 0;
        flag = 1;
    }
    else if(temp[i] > 0)
    {
        temp[i] = temp[i] - timer;
        total = total + timer;
```

```
        }
        if(temp[i] == 0 && flag == 1)
        {
            n1--;
            printf("\nProcess[%d]\t%d\t\t\t%d\t\t  %d", i + 1,burst_t[i], total -
arrival_t[i], total - arrival_t[i] - burst_t[i]);
            waiting_t = waiting_t + total - arrival_t[i] - burst_t[i];
            turnaround_t = turnaround_t + total - arrival_t[i];
            flag = 0;
        }
        if(i == n - 1)
        {
            i = 0;
        }
        else if(arrival_t[i + 1] <= total)
        {
            i++;
        }
        else
        {
            i = 0;
        }
}
    }
    avgwaiting_t = waiting_t * 1.0 / n;
    avgturnaround_t = turnaround_t * 1.0 / n;
    printf("\n\nAverage Waiting Time:\t%f", avgwaiting_t);
```

```
    printf("\nAvg Turnaround Time:\t%f\n", avgturnaround_t);
    return 0;
}
```

OUTPUT:

```
vennela@vennela-VirtualBox: ~/CSE2005
Enter the details of Processes
Arrival Time:   4
Burst Time:     4

Enter the details of Processes
Arrival Time:   1
Burst Time:     7

Enter the details of Processes
Arrival Time:   3
Burst Time:     9

Enter the details of Processes
Arrival Time:   6
Burst Time:     3

Enter Time limit:       2

ProcessID       Bursttime       TurnaroundTime   WaitingTime

Process[1]      6                       12              6
Process[2]      4                       12              8
Process[5]      3                       15              12
Process[3]      7                       25              18
Process[4]      9                       26              17

Average Waiting Time:   12.200000
Avg Turnaround Time:    18.000000
vennela@vennela-VirtualBox:~/CSE2005$
```

Question 2

Create two POSIX threads. Let the first thread copy the contents of “file1.txt” to “file2.txt” and let the second thread collapse all spaces more than one to one space in an input string. Write the main function to test the working of the two threads. Note: Use Unix system calls for file operations.

Answer:

SOURCE CODE:

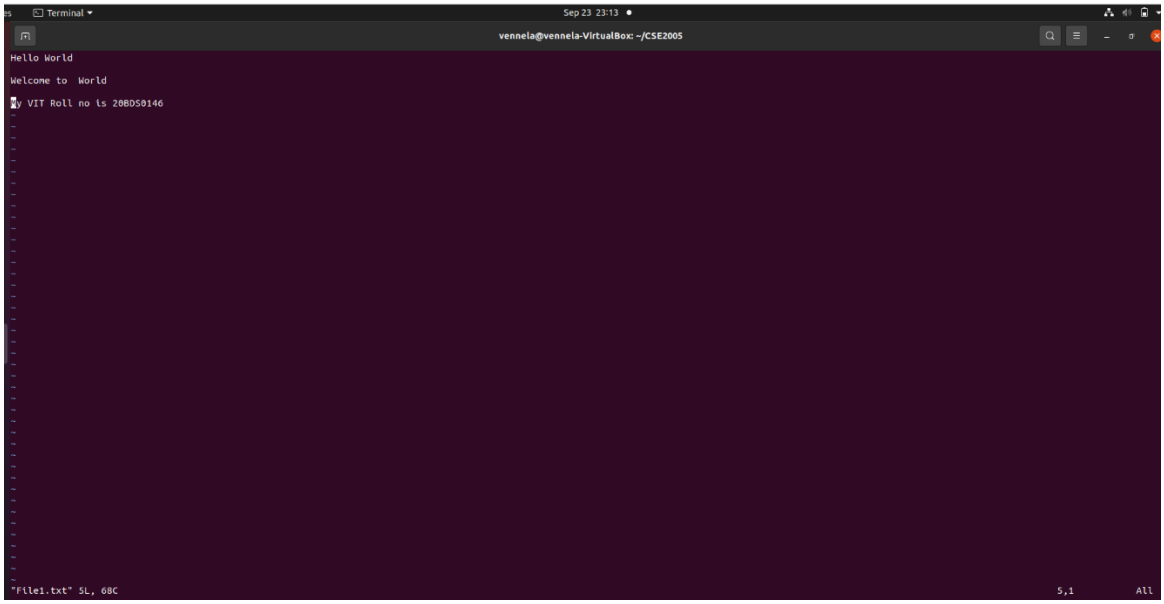
```
#include<stdio.h>

#include<unistd.h>

#include<string.h>

#include<pthread.h>

#include<stdlib.h>



void* copy_file()

{int re;

FILE *fil1,*fil2;

fil1=fopen("File1.txt","r");

fil2=fopen("File2.txt","w");

printf("\n");

while(1)

{
```

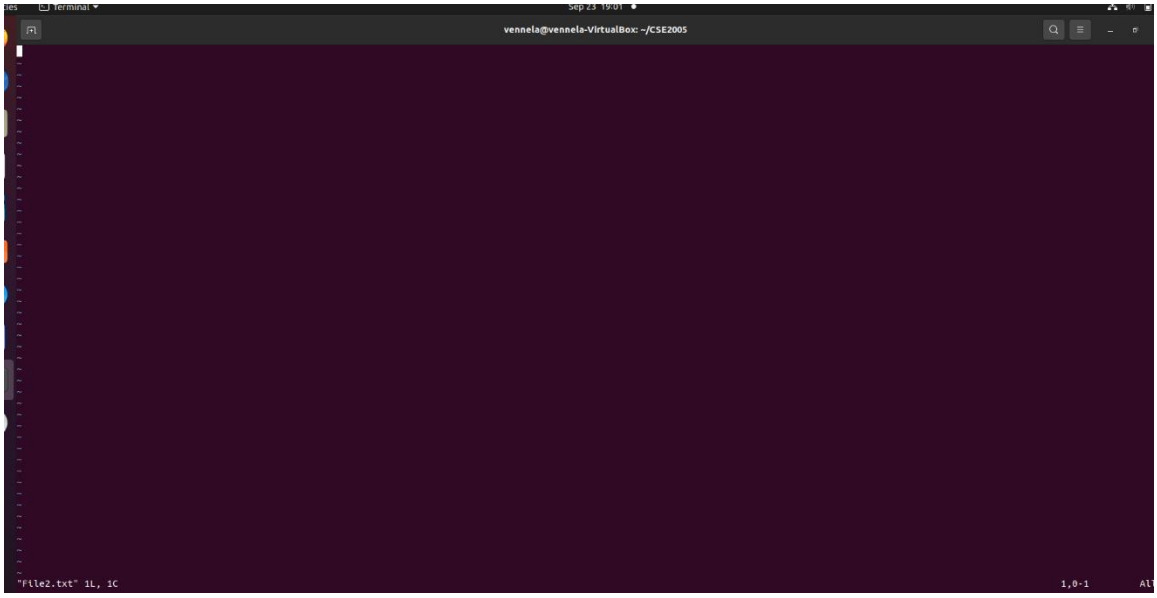
```
re=fgetc(fil1);
printf("%c",re);
if(feof(fil1)){
  break;}

fputc(re,fil2);
}
fclose(fil1);
fclose(fil2);

}

void* removal_of_space(void* arg)
{
   char string[200];
   int i, j, len;
  printf("\nInput the string for space removal -> ");
    scanf("%[^\n]s",string);

    len = strlen(string);
   for(i=0; i<len; i++) {
      if(string[0]==' ') {
         for(i=0; i<(len-1); i++)
         string[i] = string[i+1];
         string[i] = '\0';
         len--;
```

```
        i = -1;
        continue;
      }
      if(string[i]==' ' && string[i+1]==' ') {
        for(j=i; j<(len-1); j++) {
          string[j] = string[j+1];
          }
        string[j] = '\0';
        len--;
        i--;
      }
    }
    printf("\nNew String after removing extra spaces is = %s", string);
    printf("\n");

}

int main()
{pthread_t thread1,thread2;
if(pthread_create(&thread1,NULL,copy_file,NULL) !=0)
 {perror("Error in thread1 creation");
 exit(-1);
 };
if(pthread_create(&thread2,NULL,removal_of_space,NULL)!=0)
{perror("Error in thread2 creation");
 exit(-1);
```

```
 };
pthread_join(thread1,NULL);
pthread_join(thread2,NULL);
return 0;
}
```

OUTPUT:





```
vennela@vennela-VirtualBox: ~/CSE2005
vennela@vennela-VirtualBox:~/CSE2005$ gcc ex2.c -pthread
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out

Input the string for space removal ->
Hello World

Welcome to  World

My VIT Roll no is 20BDS0146
I like   Operating system  subject
◆
New String after removing extra spaces is = I like Operating system subject
vennela@vennela-VirtualBox:~/CSE2005$
```

```
Terminal                    Sep 23 23:15
vennela@vennela-VirtualBox: ~/CSE2005
Hello World

Welcome to  World

My VIT Roll no is 20BDS0146
~
~
~
```

Question 3

Write a C to implement the command given below using Unix pipes.

ls -l | grep .c$

Answer:

SOURCE CODE:

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

int main()
{
 int pid,pipeDesc[2];
 pipe(pipeDesc);
 pid=fork();

if( pid < 0 )
{
perror( "fork" );
exit( -1 );
}
```

```
 if( pid == 0 )
 {
 close(1);
 dup (pipeDesc[1]);
 close(pipeDesc[0]);
 close(pipeDesc[1]);
 execl("/bin/ls", "ls", "-l", (char *) 0);
 }

else
 {
 close(0);
 dup (pipeDesc[0]);
 close(pipeDesc[0]);
 close(pipeDesc[1]);
 execl("/usr/bin/grep","grep",".c$",(char *) 0);
 }
return 0;
}
```

OUTPUT:

```
vennela@vennela-VirtualBox: ~/CSE2005
vennela@vennela-VirtualBox:~/CSE2005$ gcc ex3.c
vennela@vennela-VirtualBox:~/CSE2005$ ./a.out
-rw-rw-r-- 1 vennela vennela   208 Sep  6 13:47 commandlineargumentsc
-rw-rw-r-- 1 vennela vennela   449 Sep 24 00:44 ex3.c
-rw-rw-r-- 1 vennela vennela    78 Sep  6 08:23 forInfinite.c
-rw-rw-r-- 1 vennela vennela   105 Sep  1 08:53 fork1.c
-rw-rw-r-- 1 vennela vennela   310 Sep  9 17:18 killExample2.c
-rw-rw-r-- 1 vennela vennela   259 Sep  9 17:33 killExample3.c
-rw-rw-r-- 1 vennela vennela   306 Sep  6 23:34 killExample.c
-rw-rw-r-- 1 vennela vennela  1215 Sep 23 13:41 matrixmultipli.c
-rw-rw-r-- 1 vennela vennela   217 Sep  9 22:18 que1.1.c
-rw-rw-r-- 1 vennela vennela   237 Sep  9 23:54 que1.c
-rw-rw-r-- 1 vennela vennela   261 Sep  6 00:22 SystemExample.c
vennela@vennela-VirtualBox:~/CSE2005$
```