Graphic Era HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

Term work

on

Operating Systems

(PCS 506)

2021-22

Submitted to:

Ms. Himadri vaidya

Submitted by:

Name:Naman Joshi

University Roll.No:2018502

ClassRoll-NO:28

Section: I

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

ACKNOWLEDGMENT

I would like to particularly thank my Operating Systems Lab Faculty Ms. Himadri vaidya for his patience, support and encouragement throughout the completion of this Term work.

At last but not the least I greatly indebted to all other persons who directly or indirectly helped me during this course.

Name: Naman Joshi
University. Roll No: 20181502

B.Tech CSE-I-V Sem

Session: 2021-22 GEHU, Dehradun

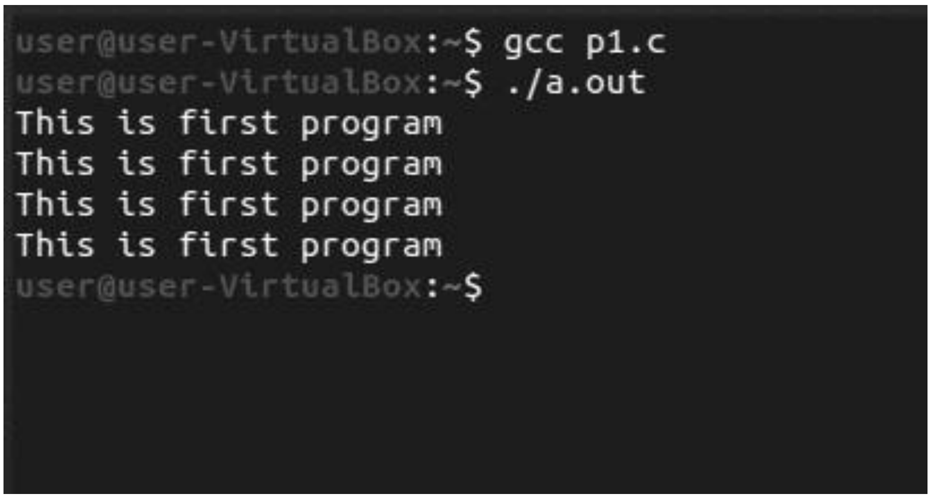
Program 1

Q1. Write a C program to demonstrate the use of fork() System call

SOURCE CODE

```
#include <stdio.h>
#include<unistd.h>
int main(){ fork();
fork();
   printf("This is first program\n");
   return 0;
}
```

OUTPUT:



```
user@user-VirtualBox:~$ gcc p1.c
user@user-VirtualBox:~$ ./a.out
This is first program
This is first program
This is first program
This is first program
user@user-VirtualBox:~$
```

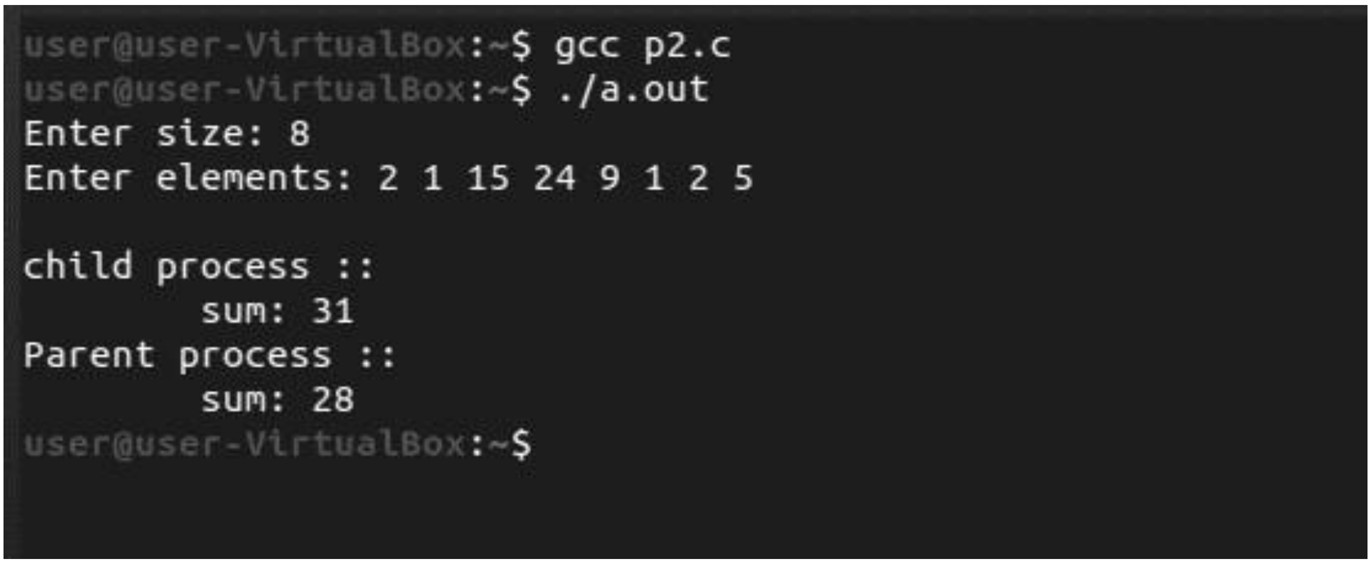
Program 2

Q2. C Program in which Parent Process Computes the SUM OF EVEN NUMBERS and Child Process Computes the sum of ODD NUMBERS stored in array using fork () . First the child process should print its answer i.e sum of odd numbers, then parent should print its answer, i.e sum of even numbers.

SOURCE CODE

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/wait.h>
int main(){
int p,n,i,sum=0;
printf("Enter size:");
scanf("%d",&n);
int arr[n];
printf("Enter elements: ");
for(i=0;i<n;i++)
scanf("%d",&arr[i]);

p=fork();
 if(p<0){
printf("failed to create child\n");
exit(1);
}
else if(p==0){
printf("\nchild process :: ");
   for(i = 0;i<n;i++)
   if(arr[i]%2!=0)
       {
          sum+=arr[i];
       }
   printf("\n\tsum: %d",sum);
} else
if(p>0){
   wait(NULL);
   printf("\nParent process :: ");
   for(i = 0;i<n;i++)
   if(arr[i]%2==0){
          sum+=arr[i];
          } printf("\n\tsum:
   %d\n",sum);
}

return 0;
}
```

OUTPUT:



```
user@user-VirtualBox:~$ gcc p2.c
user@user-VirtualBox:~$ ./a.out
Enter size: 8
Enter elements: 2 1 15 24 9 1 2 5

child process ::
        sum: 31
Parent process ::
        sum: 28
user@user-VirtualBox:~$
```

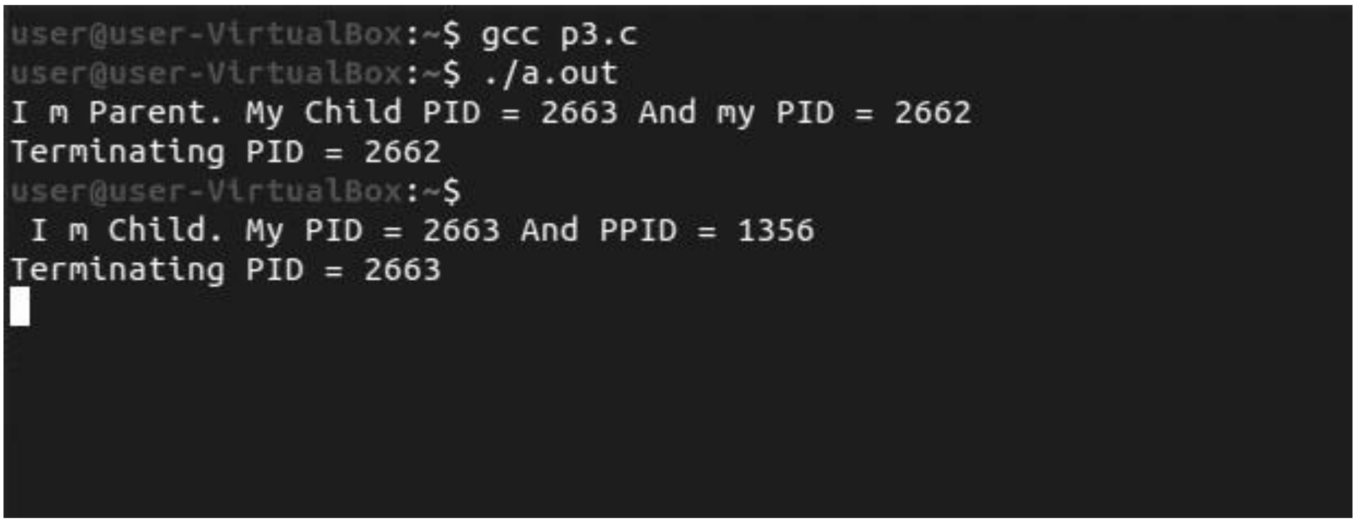
Program-3

Q3. C program to Implement the Orphan Process and Zombie Process.

SOURCE CODE

```
#include<stdio.h>
#include<unistd.>
int main(){
pid_t pid;
pid=fork();
if(pid==0){
sleep(6);
      printf("\n I m Child. My PID = %d And PPID = %d",
      getpid(),getppid());
   } else
   {
   printf("I m Parent. My Child PID = %d And my PID = %d",pid,getpid());
   }
   printf("\nTerminating PID = %d\n",getpid());
   return 0;
```

OUTPUT:



```
user@user-VirtualBox:~$ gcc p3.c
user@user-VirtualBox:~$ ./a.out
I m Parent. My Child PID = 2663 And my PID = 2662
Terminating PID = 2662
user@user-VirtualBox:~$
 I m Child. My PID = 2663 And PPID = 1356
Terminating PID = 2663
```

Program-4

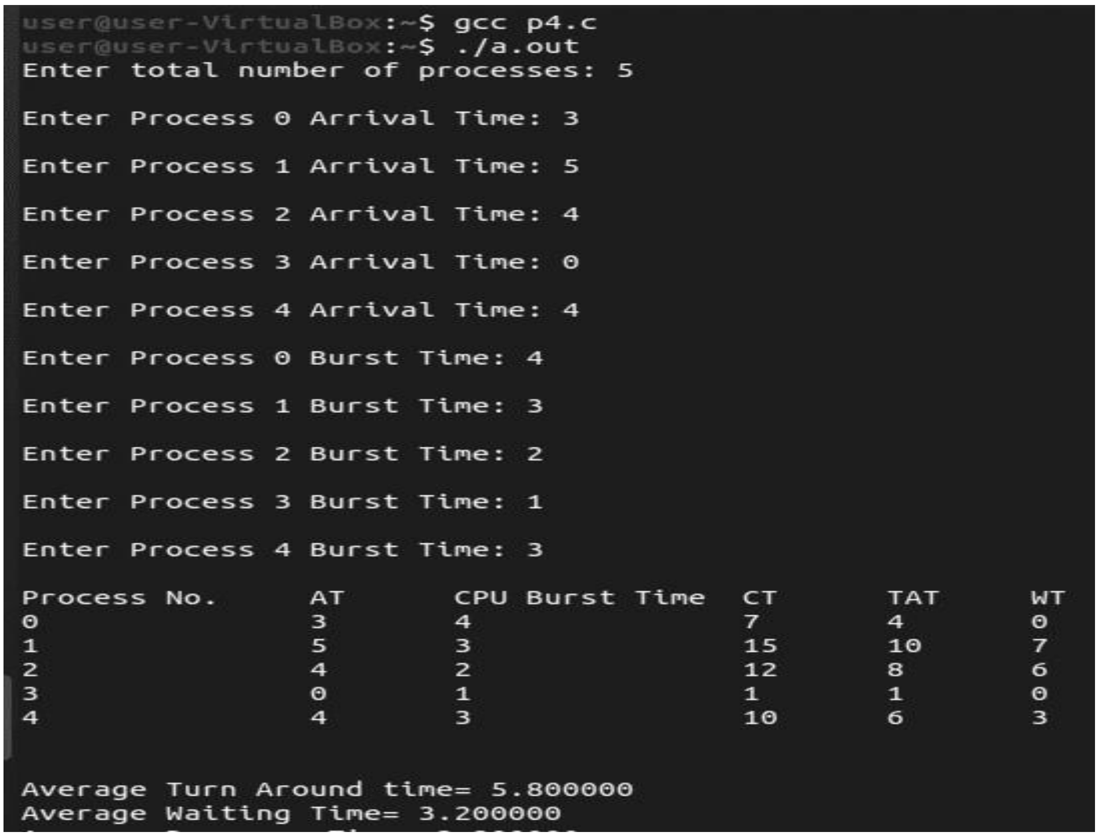
Q. C program to Implement FCFS CPU Scheduling Algorithm

SOURCE CODE

```
#include<stdio.h>
int main() {
    int n;
    printf("Enter the size of array: \n");
    scanf("%d",&n);
    int at[n];
    printf("enter the arrival time: \n");
for(int i=0;i<n;i++)     {
scanf("%d",&at[i]); }
    int bt[n];
    printf("enter the burst time: \n");
for(int i=0;i<n;i++)
    {
        scanf("%d",&bt[i]);
    }
 int ct[n],i=0;
    printf("the complition time: \n");
for(i=0;i<n;i++)
    {
        ct[i] += bt[i];

}
```

```
    printf("%d ",ct[i]);
    printf("\nthe turn around time: \n");
int tat[n],sum=0;
for(i=0;i<n;i++)
    {
 tat[i]=ct[i]-at[i];
sum += tat[i];
    }
    printf("%d ",tat[i]);
printf("\nthe waiting time: \n");
int wt[n],sum1=0;
for(i=0;i<n;i++)
    {
        wt[i]=tat[i]-bt[i];
        sum1 += wt[i];
    }
    printf("%d ",wt[i]);
   int avgtat= sum/n;
   int avgwt = sum1/n;
  printf("avg of tat: %d\n",avgtat);
printf("avg of wt: %d\n",avgwt);
}
```

<u>Output:</u>



```
user@user-VirtualBox:~$ gcc p4.c
user@user-VirtualBox:~$ ./a.out
Enter total number of processes: 5

Enter Process 0 Arrival Time: 3

Enter Process 1 Arrival Time: 5

Enter Process 2 Arrival Time: 4

Enter Process 3 Arrival Time: 0

Enter Process 4 Arrival Time: 4

Enter Process 0 Burst Time: 4

Enter Process 1 Burst Time: 3

Enter Process 2 Burst Time: 2

Enter Process 3 Burst Time: 1

Enter Process 4 Burst Time: 3

Process No.     AT      CPU Burst Time  CT      TAT     WT
0               3       4               7       4       0
1               5       3               15      10      7
2               4       2               12      8       6
3               0       1               1       1       0
4               4       3               10      6       3


Average Turn Around time= 5.800000
Average Waiting Time= 3.200000
```

Program-5

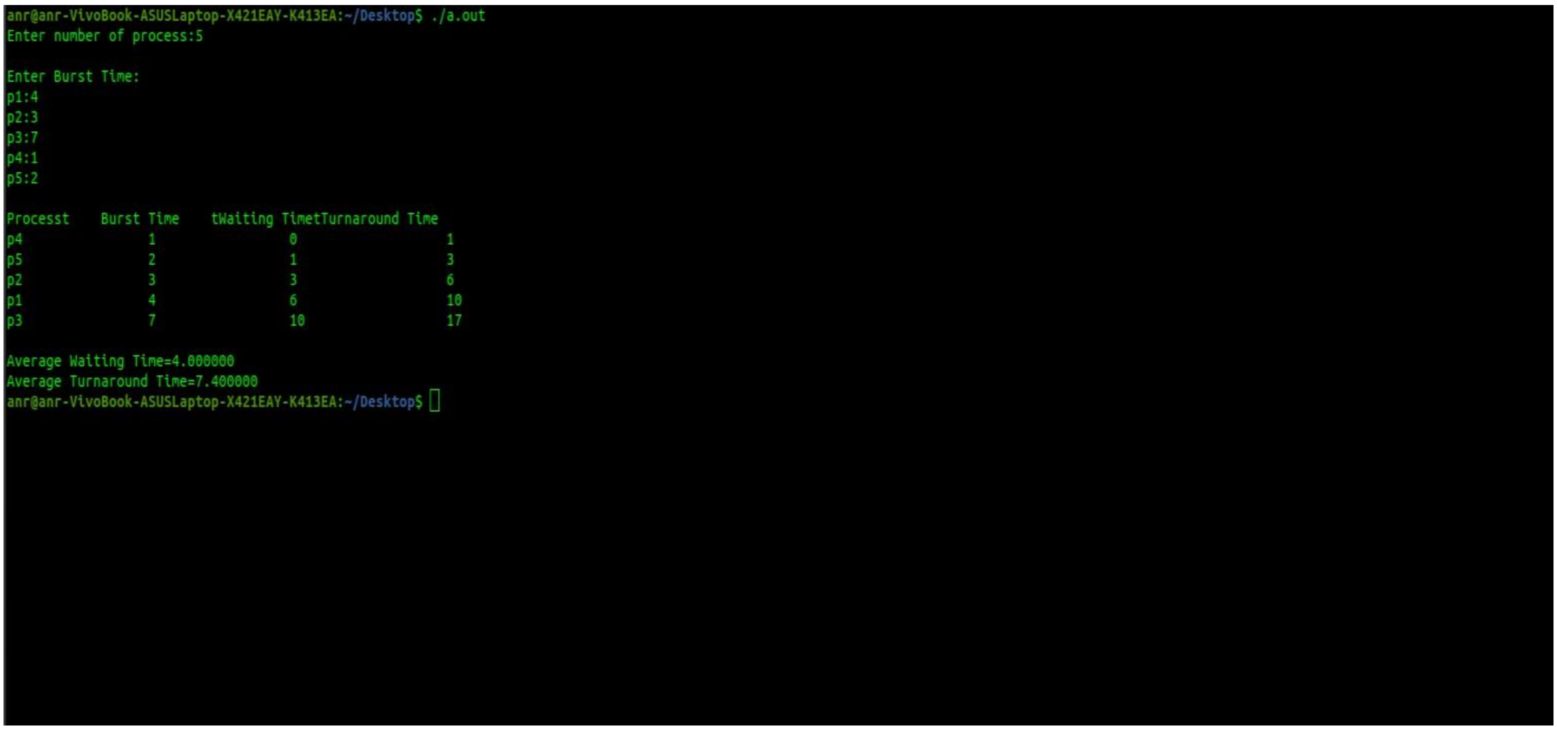
Q. C program to Implement Sortest job first non-premptive Scheduling Algorithm

SOURCE CODE

```
#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;
   }

  //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;


   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
Timet Turnaround Time");
    for(i=0;i<n;i++)
    {
         tat[i]=bt[i]+wt[i];
        total+=tat[i];
   printf("\np%d\t\t  %d\t\t    %d\t\t\
%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;

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

Output:



```
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ ./a.out
Enter number of process:5

Enter Burst Time:
p1:4
p2:3
p3:7
p4:1
p5:2

Processt    Burst Time    tWaiting TimetTurnaround Time
p4              1               0               1
p5              2               1               3
p2              3               3               6
p1              4               6               10
p3              7               10              17

Average Waiting Time=4.000000
Average Turnaround Time=7.400000
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ 
```

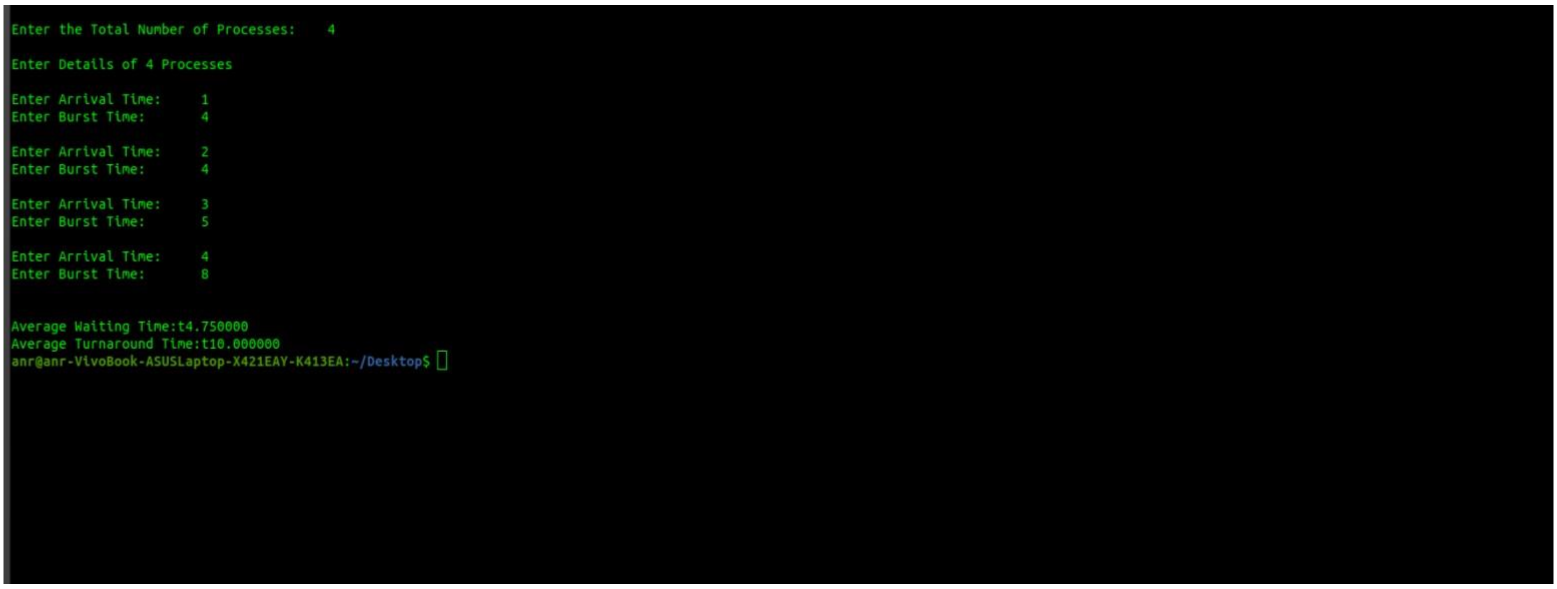
Program-6

Q. C program to Implement Sortest job first premptive Scheduling Algorithm

SOURCE CODE

```
   #include <stdio.h>

   int main()
   {
         int arrival_time[10], burst_time[10],
   temp[10];
         int i, smallest, count = 0, time, limit;
   double wait_time = 0,
   turnaround_time = 0, end;
    float average_waiting_time,
   average_turnaround_time;

printf("\nEnter the Total Number of
Processes:\t");
 scanf("%d", &limit);
 printf("\nEnter Details of %d
Processes\n", limit);
 for(i = 0; i < limit; i++)
         {
               printf("\nEnter Arrival Time:\t");
   scanf("%d", &arrival_time[i]);
   printf("Enter Burst Time:\t");
   scanf("%d", &burst_time[i]);
               temp[i] = burst_time[i];
         }
         burst_time[9] = 9999;
         for(time = 0; count != limit; time++)
         {
               smallest = 9;
               for(i = 0; i < limit; i++)
               {
                     if(arrival_time[i] <= time &&
   burst_time[i] < burst_time[smallest] &&
      burst_time[i] > 0)
                     {
                           smallest = i;
                     }
               }
               burst_time[smallest]--;
               if(burst_time[smallest] == 0)
               {
                     count++;
    end = time + 1;
    wait_time = wait_time + end -
   arrival_time[smallest] - temp[smallest];
 turnaround_time = turnaround_time + end -
arrival_time[smallest];
                }
            }
            average_waiting_time = wait_time /
    limit;
            average_turnaround_time =
    turnaround_time / limit;
    printf("\n\nAverage Waiting Time:t%lf\n",
    average_waiting_time);       printf("Average
    Turnaround Time:t%lf\n",
    average_turnaround_time);
            return 0;
    }
```

Output:



```
Enter the Total Number of Processes:    4

Enter Details of 4 Processes

Enter Arrival Time:     1
Enter Burst Time:       4

Enter Arrival Time:     2
Enter Burst Time:       4

Enter Arrival Time:     3
Enter Burst Time:       5

Enter Arrival Time:     4
Enter Burst Time:       8


Average Waiting Time:t4.750000
Average Turnaround Time:t10.000000
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ 
```

Program-7

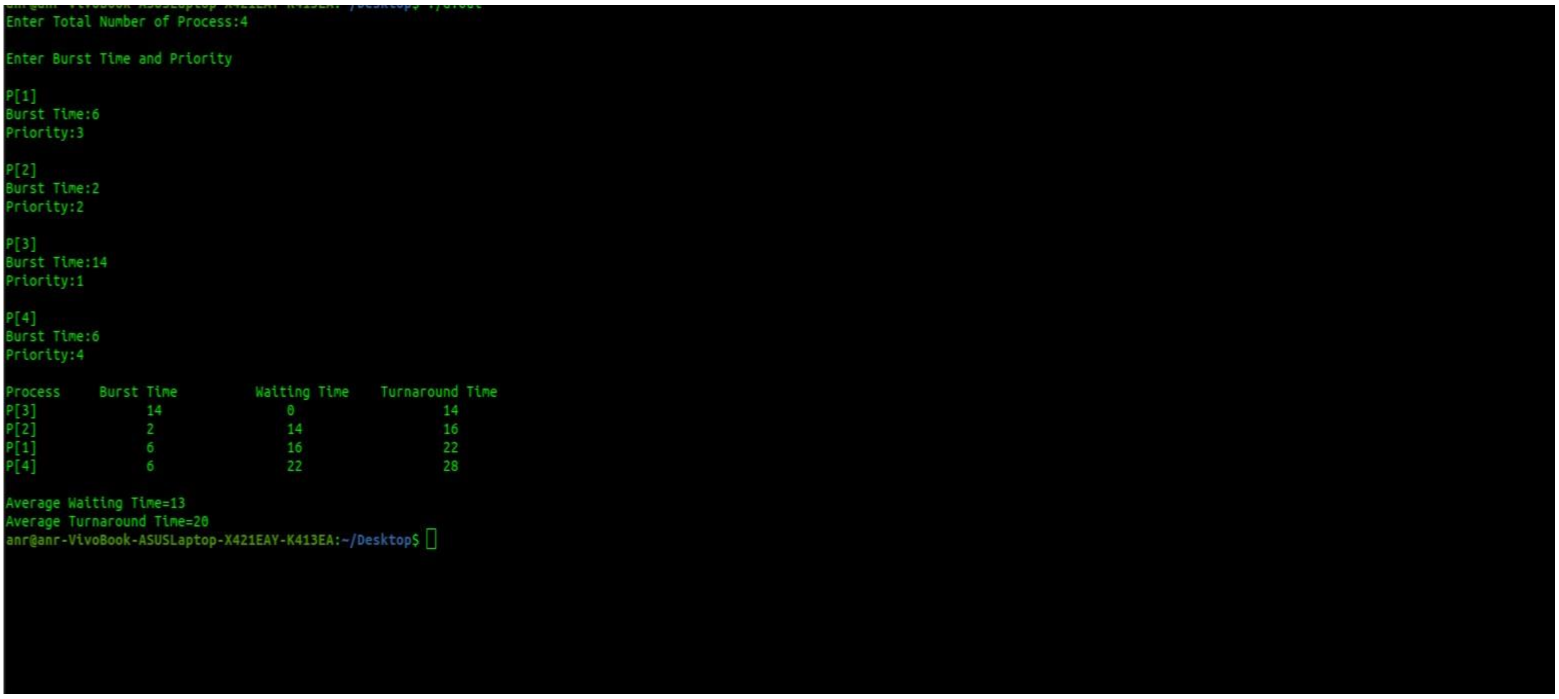
Q. C program to Implement Priority Scheduling Algorithm

SOURCE CODE

```
   #include<stdio.h>

   int main() {
   int bt[20],p[20],wt[20],
    tat[20],pr[20],i,j,n,
      total=0,pos,temp,avg_wt,avg_tat;
    printf("Enter Total Number of Process:");
   scanf("%d",&n);

     printf("\nEnter Burst Time and Priority\n");
      for(i=0;i<n;i++)
      {
      printf("\nP[%d]\n",i+1);
   printf("Burst Time:");
   scanf("%d",&bt[i]);
   printf("Priority:");
   scanf("%d",&pr[i]);
          p[i]=i+1;     //contains process number
      }

for(i=0;i<n;i++)
      {
   pos=i;
        for(j=i+1;j<n;j++)
          {
            if(pr[j]<pr[pos])
                   pos=j;
          }

          temp=pr[i];
                   pr[i]=pr[pos];
                    pr[pos]=temp;
                    temp=bt[i];
                   bt[i]=bt[pos];
                   bt[pos]=temp;
                   temp=p[i];
                   p[i]=p[pos];
                   p[pos]=temp;
      }

      wt[0]=0; //waiting time for first process is0
      //calculate waiting time
   for(i=1;i<n;i++)
      {
```

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

                total+=wt[i];
            }

            avg_wt=total/n;      //average waiting time
            total=0;

    printf("\nProcess\t   Burst Time  \tWaiting
    Time\tTurnaround Time");
     for(i=0;i<n;i++)

            {
                tat[i]=bt[i]+wt[i];     //calculate
        turnaround time
                total+=tat[i];
        printf("\nP[%d]\t\t  %d\t\t
        %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
            }

            avg_tat=total/n;
            printf("\n\nAverageWaiting
                Time=%d",avg_wt);
           printf("\nAverage Turnaround
          Time=%d\n",avg_tat);

        return 0;
}
```

Output:



```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:6
Priority:3

P[2]
Burst Time:2
Priority:2

P[3]
Burst Time:14
Priority:1

P[4]
Burst Time:6
Priority:4

Process     Burst Time          Waiting Time    Turnaround Time
P[3]              14                 0                   14
P[2]              2                  14                  16
P[1]              6                  16                  22
P[4]              6                  22                  28

Average Waiting Time=13
Average Turnaround Time=20
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ 
```

Program-8

Q. C program to Implement Round robin Scheduling Algorithm

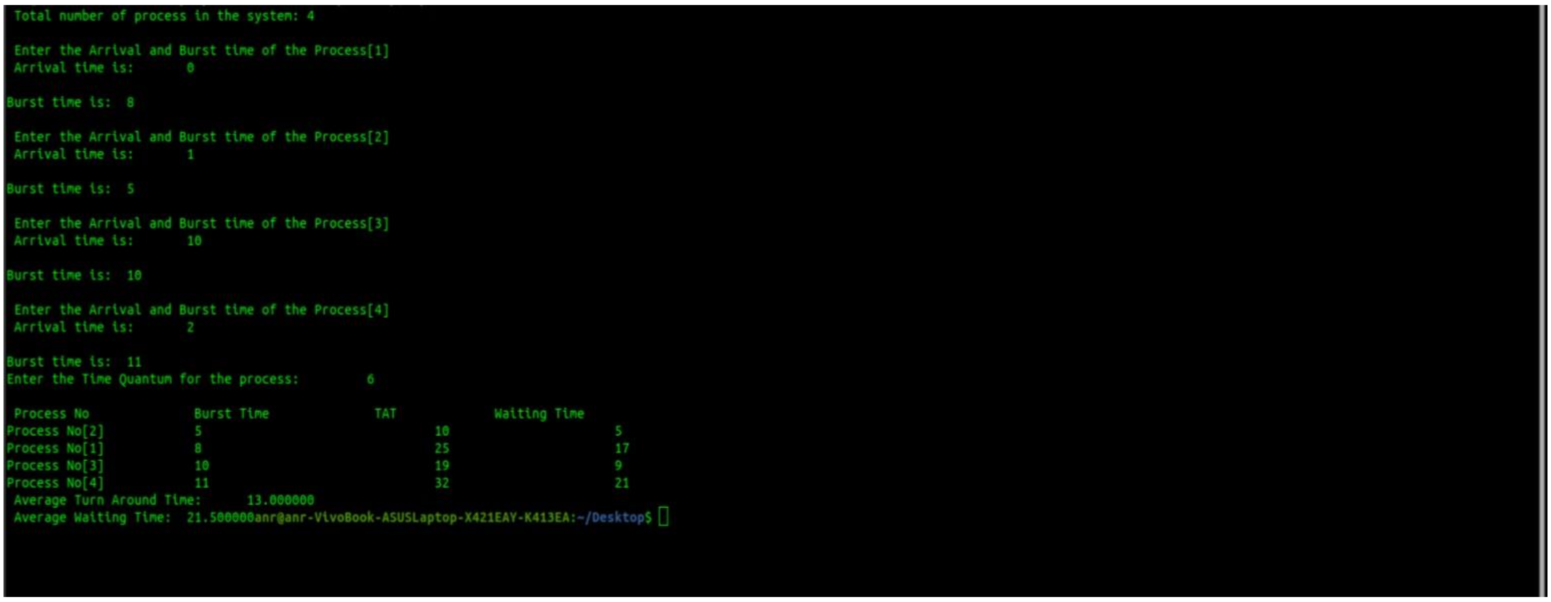
SOURCE CODE

```
#include<stdio.h>

void 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");  // Accept arrival
time
scanf("%d", &at[i]);

printf(" \nBurst time is: \t"); // Accept the
Burst time   scanf("%d", &bt[i]);

temp[i] = bt[i]; // store the burst time in temp
array
}
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) // define
the conditions
{
   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--; //decrement the process no.
printf("\nProcess No[%d] \t\t %d\t\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;
   }
```

```
}
Turn Around time   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);
}
```

Output:



```
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1

Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       10

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       2

Burst time is:  11
Enter the Time Quantum for the process:         6

 Process No             Burst Time              TAT             Waiting Time
Process No[2]           5                               10                      5
Process No[1]           8                               25                      17
Process No[3]           10                              19                      9
Process No[4]           11                              32                      21
 Average Turn Around Time:      13.000000
 Average Waiting Time:  21.500000anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ 
```

Program-9

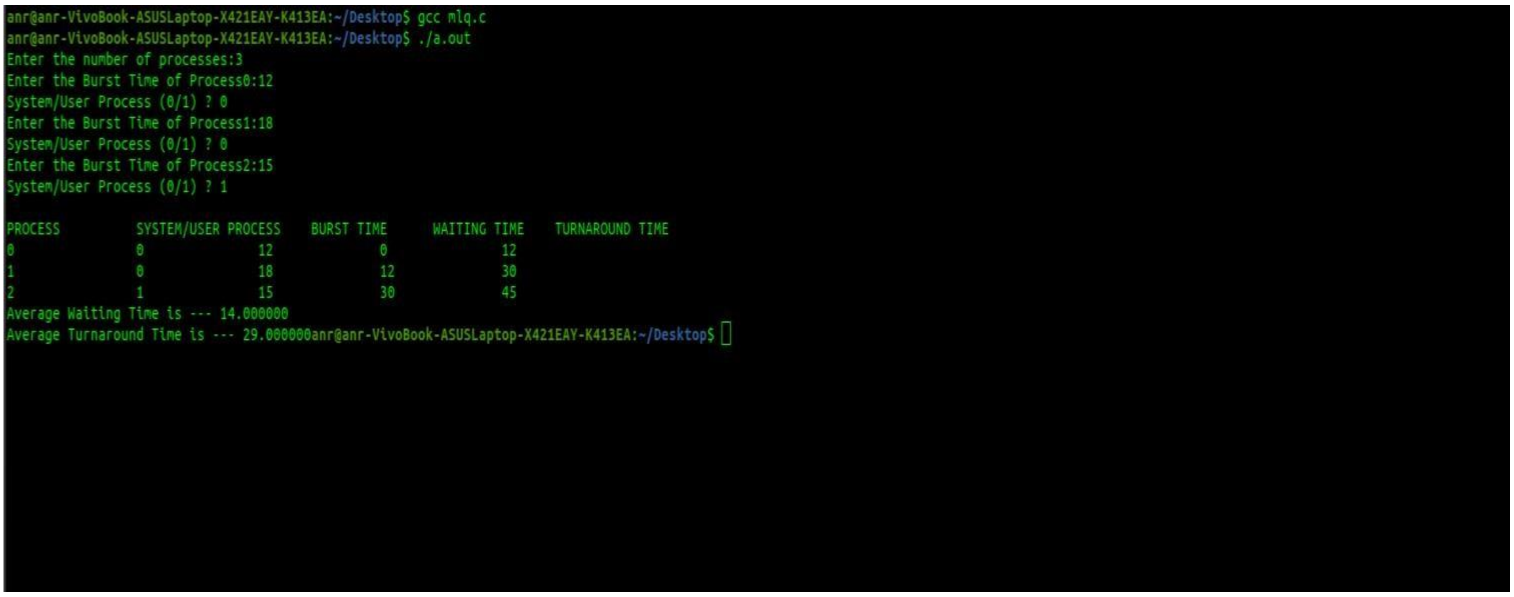
Q. C program to Implement Multilevel queue Scheduling Algorithm

SOURCE CODE

```
#include<stdio.h>
int main() {
 int p[20],bt[20], su[20], wt[20],tat[20],i, k, n,
temp;  float wtavg, tatavg;
printf("Enter the number of
                processes:");

scanf("%d",&n);
for(i=0;i<n;i++)
        {
                p[i] = i;
                printf("Enter the Burst Time of
                        Process%d:", i);
                scanf("%d",&bt[i]);
                printf("System/User Process
                        (0/1) ? ");
                scanf("%d", &su[i]);
        }
        for(i=0;i<n;i++)
                for(k=i+1;k<n;k++)
                if(su[i] > su[k])
                        {
                        temp=p[i];
                                p[i]=p[k];
                        p[k]=temp;
        temp=bt[i];
        bt[i]=bt[k];
        bt[k]=temp;
        temp=su[i];
        su[i]=su[k];
        su[k]=temp;
        }
        wtavg = wt[0] = 0;
        tatavg = tat[0] = bt[0];
        for(i=1;i<n;i++)
        {
                wt[i] = wt[i-1] + bt[i-1];
                tat[i] = tat[i-1] + bt[i];
                wtavg = wtavg + wt[i];
                tatavg = tatavg + tat[i];
        }
        printf("\nPROCESS\t\t SYSTEM/USER
PROCESS \tBURST TIME\tWAITING
TIME\tTURNAROUND TIME");
```

```
            for(i=0;i<n;i++)  {
            printf("\n%d \t\t %d \t\t %d \t\t
        %d \t\t %d ",p[i],su[i],bt[i],wt[i],tat[i]);
            printf("\nAverage Waiting Time is ---
        %f",wtavg/n);
            printf("\nAverage Turnaround Time is -
        -- %f",tatavg/n);
            return 0;
        }
    }
```

Output:



```
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ gcc mlq.c
anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ ./a.out
Enter the number of processes:3
Enter the Burst Time of Process0:12
System/User Process (0/1) ? 0
Enter the Burst Time of Process1:18
System/User Process (0/1) ? 0
Enter the Burst Time of Process2:15
System/User Process (0/1) ? 1

PROCESS         SYSTEM/USER PROCESS    BURST TIME      WAITING TIME    TURNAROUND TIME
0               0               12              0               12
1               0               18              12              30
2               1               15              30              45
Average Waiting Time is --- 14.000000
Average Turnaround Time is --- 29.000000anr@anr-VivoBook-ASUSLaptop-X421EAY-K413EA:~/Desktop$ 
```