CSE-316 OPERATING SYSTEMS

SIMULATION PROJECT

ASSIGNMENT QUESTIONS – 08,19

NAME : V.HEMANTH REDDY

REGD. : 11607482

SECTION : K1654 B37

EMAIL : vankayalahemanth@gmail.com

SUBMITTING TO:

RADHIKA SHARMA

REMARKS

_____

_

Question:08

code:

```
 #include<stdio.h>

 int main()
{
  int process,resource,i,j,instanc,k=0,count1=0,count2=0; //count,k      variables are taken for counting
purpose
  printf("\n\t Enter No. of Process:-\n");
  printf("\t\t");
  scanf("%d",&process);                          //Entering No. of Processes
  printf("\n\tEnter No. of Resources:-\n");
  printf("\t\t");
  scanf("%d",&resource);                      //No. of Resources

  int avail[resource],max[process][resource],allot[process][resource],need[process]
[resource],completed[process];

  for(i=0;i<process;i++)
  completed[i]=0;                             //Setting Flag for uncompleted Process

  printf("\n\tEnter No. of Available Instances\n");

  for(i=0;i<resource;i++)
  {
   printf("\t\t");
   scanf("%d",&instanc);
   avail[i]=instanc;                        // Storing Available instances
  }

  printf("\n\tEnter Maximum No. of instances of resources that a Process need:\n");

  for(i=0;i<process;i++)
  {
   printf("\n\t For P[%d]",i);
   for(j=0;j<resource;j++)
    {
      printf("\t");
      scanf("%d",&instanc);
      max[i][j]=instanc;
    }
  }
  printf("\n\t Enter no. of instances already allocated to process of a resource:\n");

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

```
  {
    printf("\n\t For P[%d]\t",i);
    for(j=0;j<resource;j++)
     {
        printf("\t\t");
        scanf("%d",&instanc);
        allot[i][j]=instanc;
        need[i][j]=max[i][j]-allot[i][j];       //calculating Need of each process
     }
 }
printf("\n\t Safe Sequence is:- \t");

 while(count1!=process)
 {
  count2=count1;
  for(i=0;i<process;i++)
  {
    for(j=0;j<resource;j++)
    {
       if(need[i][j]<=avail[j])
         {
          k++;
         }
    }
    if(k==resource && completed[i]==0 )
     {
       printf("P[%d]\t",i);
       completed[i]=1;
 for(j=0;j<resource;j++)
        {
          avail[j]=avail[j]+allot[i][j];
         }
        count1++;
     }
     k=0;
   }

     if(count1==count2)
     {
     printf("\t\t Stop ..After this.....Deadlock \n");
     break;
   }
 }
 return 0;
}
```

DESCRIPTION

Bankers Algrithm:Bankers Algorithm,sometimes referred to as the detection algorithm.By

simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

For the Banker's algorithm to work, it needs to know three things:

- How much of each resource each process could possibly request[MAX]
- How much of each resource each process is currently holding[ALLOCATED]
- How much of each resource the system currently has available[AVAILABLE]

ALGORITHM FOR BANKER'S

Let 'n' be the number of processes in the system and 'm' be the number of resources types.

Available :

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available[j] = k means there are **'k'** instances of resource type R_j

Max :

- It is a 2-d array of size '**n*m**' that defines the maximum demand of each process in a system.
- Max[i, j] = k means process P_i may request at most **'k'** instances of resource type R_j.

Allocation :

- It is a 2-d array of size **'n*m'** that defines the number of resources of each type currently allocated to each process.
- Allocation[i, j] = k means process P_i is currently allocated **'k'** instances of resource type R_j

Need :

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [i, j] = k means process P_i currently allocated **'k'** instances of resource type R_j
- Need [i, j] = Max [i, j] – Allocation [i, j]

CONSTRAINTS

1. To calculate available from the given number of resources

```
printf("\n\nEnter the Available Resources : ");
for(i = 0; i < r; i++)
   scanf("%d", &avail[i]);
```

2. To calculate maximum form the given number of processes and resources

```
printf("\n\nEnter the Max Matrix for each process : ");
   for(i = 0; i < p; i++)
   {
      printf("\nFor process %d : ", i + 1);
      for(j = 0; j < r; j++)
         scanf("%d", &Max[i][j]);
   }
```

3. To calculate available form the given process and resources

```
   printf(“\n\nEnter the allocation for each process:”);
    for(i=0;i<p;i++)
{
printf(“\nFor process%d:”i+1);
for(j=0;j<r;j++)
scanf(“%d”,&alloc[i][j]);
}
```

4. To calculate need from given process and resources

```
      for(i=0;i<p;i++)
{
  for(j=0;j<r;j++)
{
need[i][j]=Max[i][j]-alloc[i][j];
}
}
```

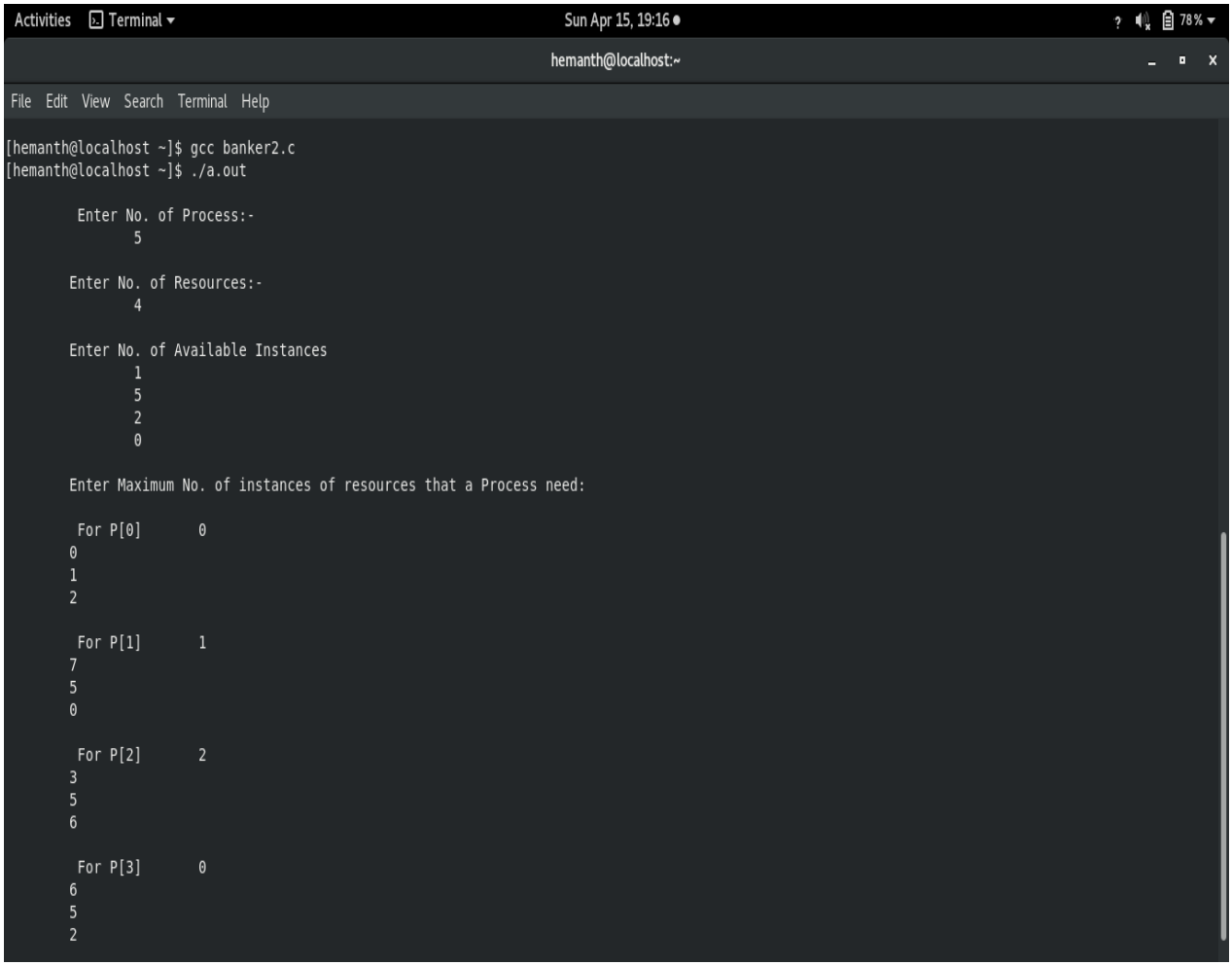
COMPLEXITY

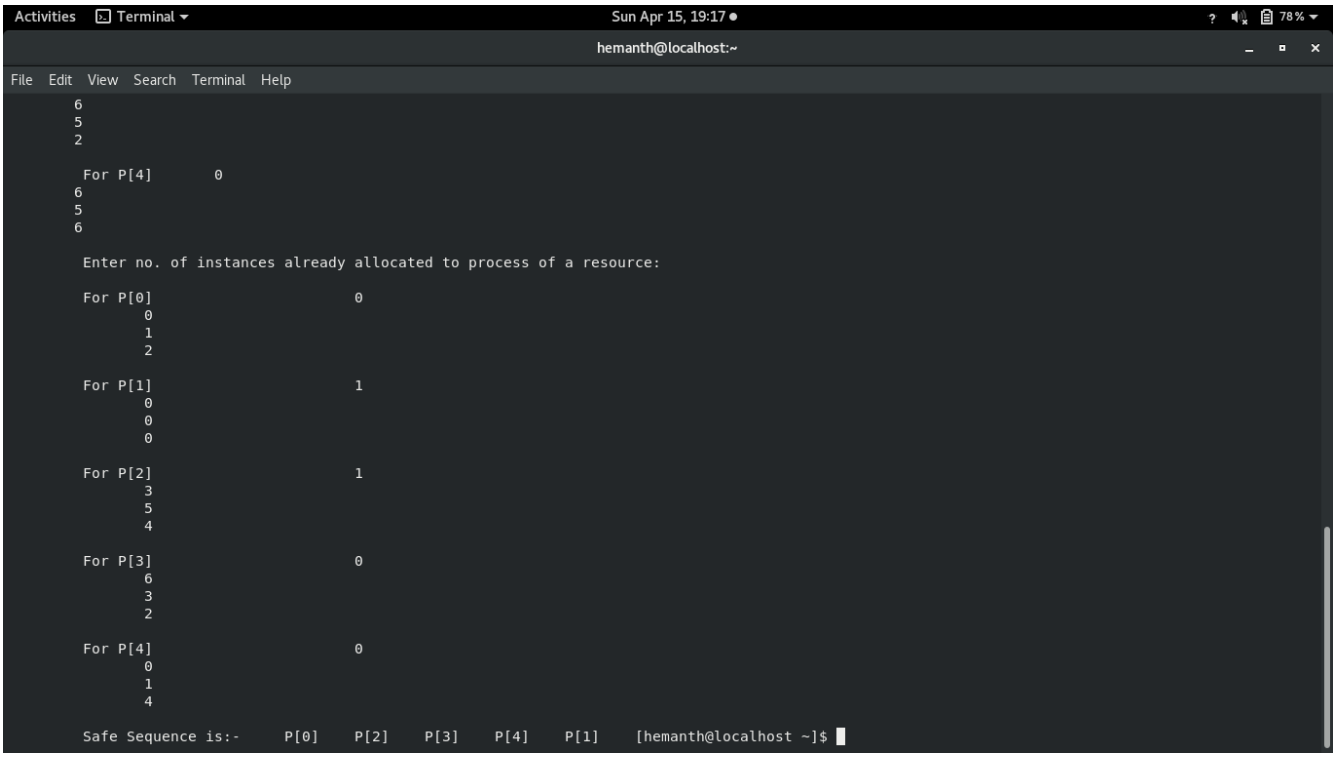
Lines	Complexity
Total Complexity:	O(n*m)

TEST CASES

PROCESS		ALLOCATION			MAX					NEED		
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	0	0	0	0
P1	1	0	0	0	1	7	5	0	0	7	5	0
P2	1	3	5	4	2	3	5	6	1	0	0	2
P3	0	6	3	2	0	6	5	2	0	0	2	0
P4	0	0	1	4	0	6	5	6	0	5	1	6

AVAILABLE			
A	B	C	D
1	5	2	0


Activities
Terminal
Sun Apr 15, 19:16
78%
hemanth@localhost:~
File Edit View Search Terminal Help
[hemanth@localhost ~]$ gcc banker2.c
[hemanth@localhost ~]$ ./a.out
Enter No. of Process:-
5
Enter No. of Resources:-
4
Enter No. of Available Instances
1
5
2
0
Enter Maximum No. of instances of resources that a Process need:
For P[0] 0
0
1
2
For P[1] 1
7
5
0
For P[2] 2
3
5
6
For P[3] 0
6
5
2



➔ SAFE SEQUENCE is:P[0] P[2] P[3] P[4] P[1]
➔ The System is in Safe State.

GITHUB LINK:
https://github.com/Vhr5196/os-assigment-k1654-B37

Question-19

Code:

```
#include<stdio.h>

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

    printf("\nEnter Burst Time and Priority of student\n");
    for(i=0;i<n;i++)
    {
        printf("\nst[%d]\n",i+1);
        printf("Burst Time of each student:");
        scanf("%d",&bt[i]);
        printf("Priority of each student:");
        scanf("%d",&pr[i]);
        st[i]=i+1;          //contains students number
    }

    //sorting burst time, priority and process number in ascending
order using selection sort
    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=st[i];
        st[i]=st[pos];
        st[pos]=temp;
    }

    wt[0]=0;    //waiting time for first student is zero

    //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("\nStudent\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("\nst[%d]\t\t  %d\t\t    %d\t\t\t%d",st[i],bt[i],wt[i],tat[i]);
}
  avg_tat=total/n;

  printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\nAverage Turnaround Time=%d\n",avg_tat);
    printf("Waiting time of students get minimised:");
    return 0;
}
```

Description:

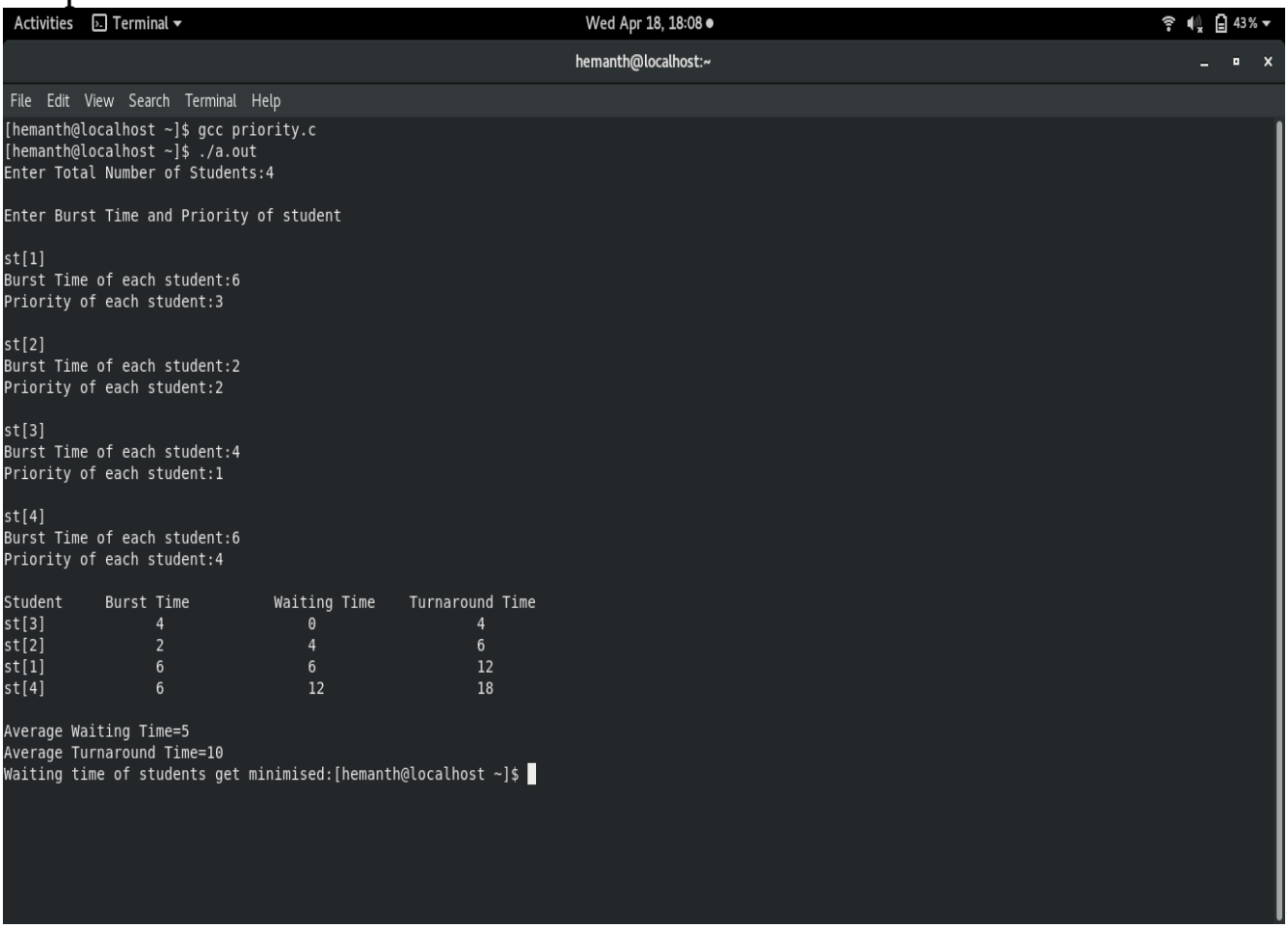
Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Implementation :

```
1- First input the processes with their burst time
   and priority.
2- Sort the processes, burst time and priority
   according to the priority.
3- Now simply apply FCFS algorithm.
```

Output:


Activities
Terminal
Wed Apr 18, 18:08
43%
hemanth@localhost:~
File Edit View Search Terminal Help
[hemanth@localhost ~]$ gcc priority.c
[hemanth@localhost ~]$ ./a.out
Enter Total Number of Students:4
Enter Burst Time and Priority of student
st[1]
Burst Time of each student:6
Priority of each student:3
st[2]
Burst Time of each student:2
Priority of each student:2
st[3]
Burst Time of each student:4
Priority of each student:1
st[4]
Burst Time of each student:6
Priority of each student:4
Student Burst Time Waiting Time Turnaround Time
st[3] 4 0 4
st[2] 2 4 6
st[1] 6 6 12
st[4] 6 12 18
Average Waiting Time=5
Average Turnaround Time=10
Waiting time of students get minimised:[hemanth@localhost ~]$