OPERATING SYSTEMS

Course Code : SWE3001 Slot : F2+TF2

Lab Assignment - 5

Name : S.Deepan

Reg No : 19MIS0102

Bankers algorithm (use below data as input for your program):

Available

A	B	C	D
6	3	5	4

Process	Current allocation				Maximum demand			
	A	B	C	D	A	B	C	D
P0	2	0	2	1	9	5	5	5
P1	0	1	1	1	2	2	3	3
P2	4	1	0	2	7	5	4	4
P3	1	0	0	1	3	3	3	2
P4	1	1	0	0	5	2	2	1
P5	1	0	1	1	4	4	4	4

Code :

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

void print(int x[][10],int n,int m)
{
        int i,j;
        for(i=0;i<n;i++)
        {
```

```
                printf("\n");
                for(j=0;j<m;j++)
                {
                        printf("%d\t",x[i][j]);
                }
        }
}
void res_req(int allot[10][10],int need[10][10],int avail[10][10],int pid,int m)
//create Resource Request algorithm function
{
        int reqmat[1][10]; //declaration of request matrix
        int i;
        printf("\n Enter additional request :- \n");
        //If the user want to make any additional requests, they must provide input.
        for(i=0;i<m;i++)
        {
        printf(" Request for resource %d : ",i+1);
        //input from the user of their resource request
        scanf("%d",&reqmat[0][i]);
        //scan request matrix
        }
        for(i=0;i<m;i++)
                if(reqmat[0][i] > need[pid][i])
        //condition for checking if request matrix is greater than need matrix
        {
                printf("\n Error encountered.\n");
                exit(0);
        }
        for(i=0;i<m;i++)
        if(reqmat[0][i] > avail[0][i])
```

```
        //If the request matrix exceeds the available matrix, the resource is
        unavailable.
        {
                printf("\n Resources unavailable.\n");
                exit(0);
        }
        for(i=0;i<m;i++)
        {
                avail[0][i]-=reqmat[0][i]; //avail-reqmat
                allot[pid][i]+=reqmat[0][i];//allot+reqmat
                need[pid][i]-=reqmat[0][i];//need-reqmat
        }
}
int safety(int allot[][10],int need[][10],int avail[1][10],int n,int m,int a[])
//create Safety algorithm function
{
        int i,j,k,x=0;
        int finish[10];
        int work[1][10];
        //makes a duplicate of the resources that are currently available
        int pflag=0,flag=0;
        for(i=0;i<n;i++)
        finish[i]=0; // Marking tha processes as not finished
        for(i=0;i<m;i++)
        work[0][i]=avail[0][i];
        for(k=0;k<n;k++)
        {
                for(i=0;i<n;i++)
                {
                        if(finish[i] == 0) //check if a process is finished
```

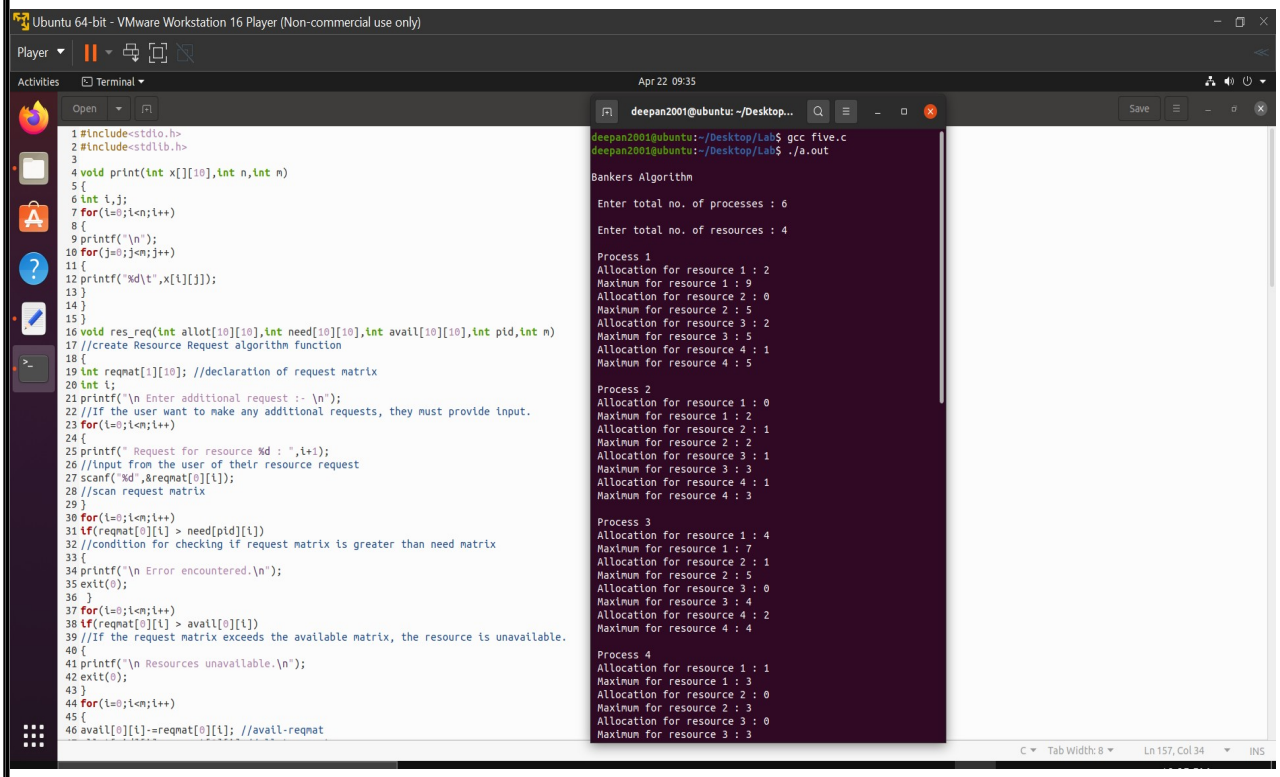
```
                {
                        flag=0;
                        // Find a process that isn't done but whose requirements
                        can be met using present resources.
                        for(j=0;j<m;j++)
                        {
                                if(need[i][j] > avail[0][j])
                                //condition if the resources of current process is less
                                than work
                                flag=1;
                        }
                        if(flag == 0 && finish[i] == 0)
                        // If all needs of process are satisfied.
                        {
                        for(j=0;j<m;j++)
                        avail[0][j]+=allot[i][j];
                        // Add the current Process's assigned resources to the
                        available resources.
                        finish[i]=1; //mark process as finished
                        pflag++; // Adding the process to safe sequence.
                        a[x++]=i;
                        }
                }
        }
        if(pflag == n)
        return 1;
    }
    return 0;
}
```

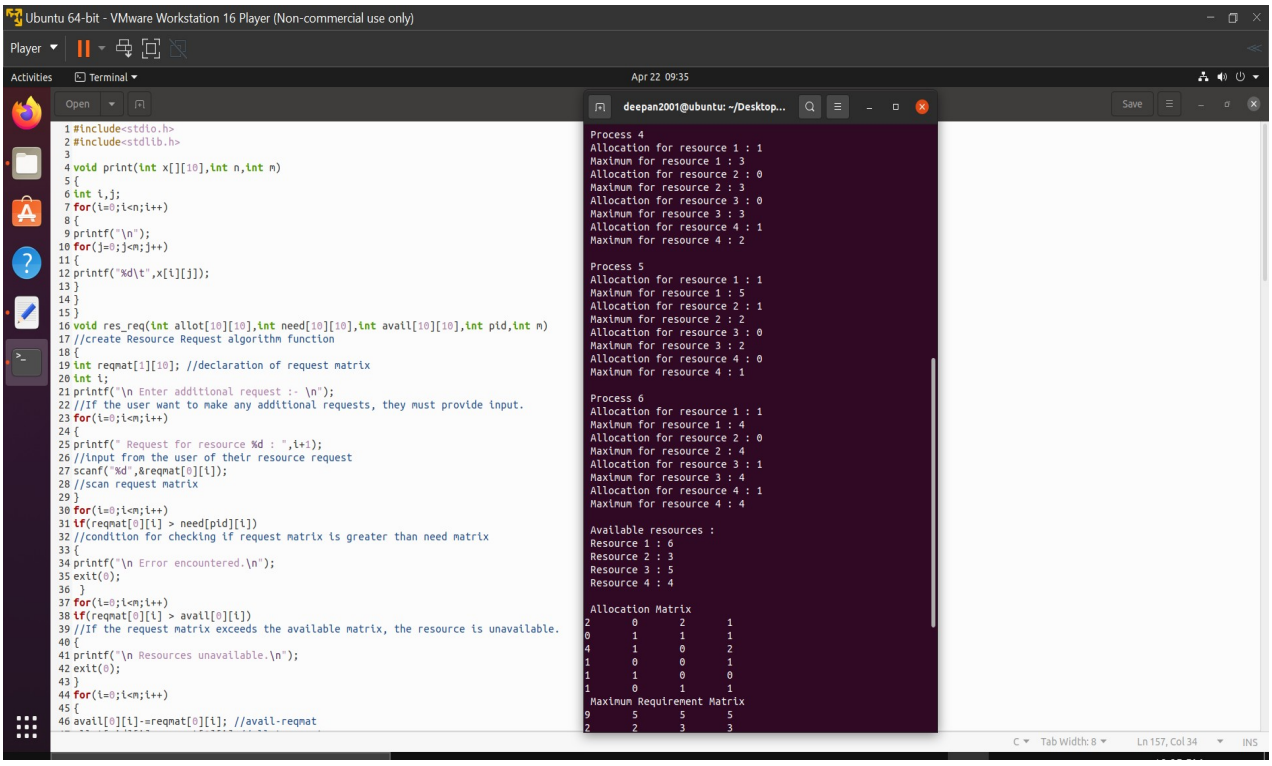
```
void accept(int allot[][10],int need[][10],int maxmat[10][10],int avail[1][10],int
*n,int *m)
//create Banker's Algorithm function
{
        int i,j;
        printf("\n Enter total no. of processes : ");
        scanf("%d",n);
        printf("\n Enter total no. of resources : ");
        scanf("%d",m);
        for(i=0;i<*n;i++)
        {
                printf("\n Process %d\n",i+1);
                for(j=0;j<*m;j++)
                {
                        printf(" Allocation for resource %d : ",j+1);
                        scanf("%d",&allot[i][j]);
                        printf(" Maximum for resource %d : ",j+1);
                        scanf("%d",&maxmat[i][j]);
                }
        }
        printf("\n Available resources : \n");
        for(i=0;i<*m;i++)
        {
                printf(" Resource %d : ",i+1); //enter the available resources
                scanf("%d",&avail[0][i]);
        }
        for(i=0;i<*n;i++)
        for(j=0;j<*m;j++)
        need[i][j]=maxmat[i][j]-allot[i][j];
        //Need of instance = max instance assigned instance
```

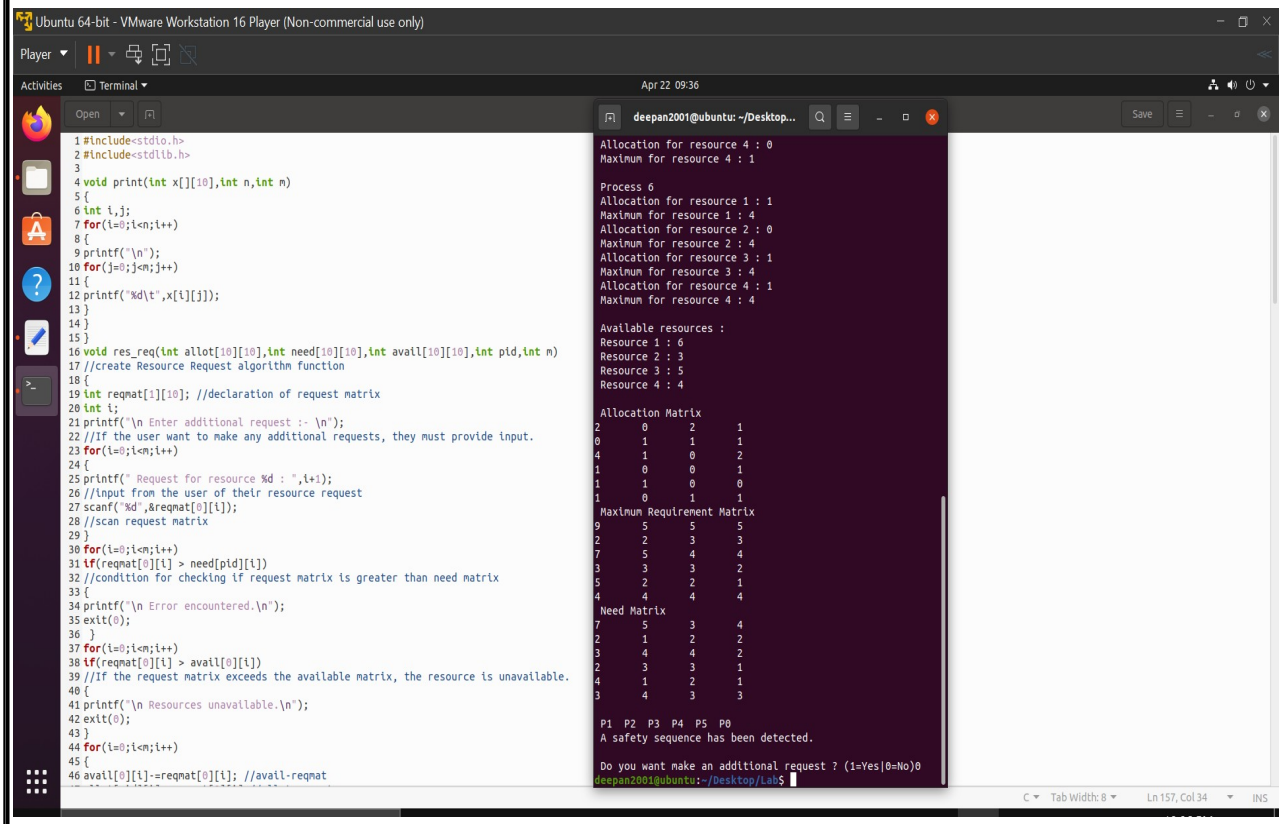
```
        printf("\n Allocation Matrix");
        print(allot,*n,*m);
        printf("\n Maximum Requirement Matrix");
        print(maxmat,*n,*m);
        printf("\n Need Matrix");
        print(need,*n,*m);
}
int banker(int allot[][10],int need[][10],int avail[1][10],int n,int m)
{
        int j,i,a[10];
        j=safety(allot,need,avail,n,m,a);
        //safety function call and store the output in j
        if(j != 0 )
        {
                printf("\n\n");
                for(i=0;i<n;i++)
                printf(" P%d ",a[i]); //display of safe sequence
                printf("\n A safety sequence has been detected.\n"); //system is in safe
                state
                return 1;
        }
        else
        {
        printf("\n Deadlock has occured.\n"); //if it is not a safety sequence then print
        this
        return 0;
        }
}

int main()
```

```
{
        printf("\nBankers Algorithm\n");
        int ret; int allot[10][10];
        int maxmat[10][10];
        int need[10][10];
        int avail[1][10];
        int n,m,pid,ch;
        accept(allot,need,maxmat,avail,&n,&m);
        ret=banker(allot,need,avail,n,m);
        if(ret !=0 ) //If there isn't a deadlock, make a resource request.
        {
                printf("\n Do you want make an additional request ? (1=Yes|
                0=No)");
                //If the user wants a resource request, ask for their input
                scanf("%d",&ch);
                if(ch == 1)
                {
                        printf("\n Enter process no. : ");
                        scanf("%d",&pid);
                        res_req(allot,need,avail,pid-1,m);
                        ret=banker(allot,need,avail,n,m);
                        if(ret == 0 )
                        exit(0);}
                }
                else
                exit(0);
        return 0;
}
```

Output :


Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Apr 22 09:35
Open
Save
deepan2001@ubuntu: ~/Desktop...
deepan2001@ubuntu:~/Desktop/Lab$ gcc five.c
deepan2001@ubuntu:~/Desktop/Lab$ ./a.out
Bankers Algorithm
Enter total no. of processes : 6
Enter total no. of resources : 4
Process 1
Allocation for resource 1 : 2
Maximum for resource 1 : 9
Allocation for resource 2 : 0
Maximum for resource 2 : 5
Allocation for resource 3 : 2
Maximum for resource 3 : 5
Allocation for resource 4 : 1
Maximum for resource 4 : 5
Process 2
Allocation for resource 1 : 0
Maximum for resource 1 : 2
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 1
Maximum for resource 3 : 3
Allocation for resource 4 : 1
Maximum for resource 4 : 3
Process 3
Allocation for resource 1 : 4
Maximum for resource 1 : 7
Allocation for resource 2 : 1
Maximum for resource 2 : 5
Allocation for resource 3 : 0
Maximum for resource 3 : 4
Allocation for resource 4 : 2
Maximum for resource 4 : 4
Process 4
Allocation for resource 1 : 1
Maximum for resource 1 : 3
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 0
Maximum for resource 3 : 3
C
Tab Width: 8
Ln 157, Col 34
INS


Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Apr 22 09:35
Open
Save
deepan2001@ubuntu: ~/Desktop...
Process 4
Allocation for resource 1 : 1
Maximum for resource 1 : 3
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 0
Maximum for resource 3 : 3
Allocation for resource 4 : 1
Maximum for resource 4 : 2
Process 5
Allocation for resource 1 : 1
Maximum for resource 1 : 5
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 0
Maximum for resource 3 : 2
Allocation for resource 4 : 0
Maximum for resource 4 : 1
Process 6
Allocation for resource 1 : 1
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 4
Allocation for resource 3 : 1
Maximum for resource 3 : 4
Allocation for resource 4 : 1
Maximum for resource 4 : 4
Available resources :
Resource 1 : 6
Resource 2 : 3
Resource 3 : 5
Resource 4 : 4
Allocation Matrix
2 0 2 1
0 1 1 1
4 1 0 2
1 0 0 1
1 1 0 0
1 0 1 1
Maximum Requirement Matrix
9 5 5 5
2 2 3 3
C
Tab Width: 8
Ln 157, Col 34
INS


Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Apr 22 09:36
Open
Save
deepan2001@ubuntu: ~/Desktop...
#include<stdio.h>
#include<stdlib.h>
void print(int x[][10],int n,int m)
{
int i,j;
for(i=0;i<n;i++)
{
printf("\n");
for(j=0;j<m;j++)
{
printf("%d\t",x[i][j]);
}
}
}
void res_req(int allot[10][10],int need[10][10],int avail[10][10],int pid,int m)
//create Resource Request algorithm function
{
int reqmat[1][10]; //declaration of request matrix
int i;
printf("\n Enter additional request :- \n");
//If the user want to make any additional requests, they must provide input.
for(i=0;i<m;i++)
{
printf(" Request for resource %d : ",i+1);
//input from the user of their resource request
scanf("%d",&reqmat[0][i]);
//scan request matrix
}
for(i=0;i<m;i++)
if(reqmat[0][i] > need[pid][i])
//condition for checking if request matrix is greater than need matrix
{
printf("\n Error encountered.\n");
exit(0);
}
for(i=0;i<m;i++)
if(reqmat[0][i] > avail[0][i])
//If the request matrix exceeds the available matrix, the resource is unavailable.
{
printf("\n Resources unavailable.\n");
exit(0);
}
for(i=0;i<m;i++)
{
avail[0][i]-=reqmat[0][i]; //avail-reqmat
Allocation for resource 4 : 0
Maximum for resource 4 : 1
Process 6
Allocation for resource 1 : 1
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 4
Allocation for resource 3 : 1
Maximum for resource 3 : 4
Allocation for resource 4 : 1
Maximum for resource 4 : 4
Available resources :
Resource 1 : 6
Resource 2 : 3
Resource 3 : 5
Resource 4 : 4
Allocation Matrix
2 0 2 1
0 1 1 1
4 1 0 2
1 0 0 1
1 1 0 0
1 0 1 1
Maximum Requirement Matrix
9 5 5 5
2 2 3 3
7 5 4 4
3 3 3 2
5 2 2 1
4 4 4 4
Need Matrix
7 5 3 4
2 1 2 2
3 4 4 2
2 3 3 1
4 1 2 1
3 4 3 3
P1 P2 P3 P4 P5 P0
A safety sequence has been detected.
Do you want make an additional request ? (1=Yes|0=No)0
deepan2001@ubuntu:~/Desktop/Lab$
C
Tab Width: 8
Ln 157, Col 34
INS