Name: Om Jadhav **Roll No.:** I3275 **Div:** 2

Statement: Implement the C program for Deadlock Avoidance Algorithm: Bankers Algorithm.

Code:

```
#include<stdio.h>
void main()
{
	int allocated[20][20],max[20][20],available[20]={0},need[20][20],total[20];
	int finish[20]={0};
	int cntProcess,cntRes,process,res,flag,execFlag,executed;

	printf("Enter no. of processes: ");
	scanf("%d",&process);
	printf("\nEnter no. of resources: ");
	scanf("%d",&res);

	printf("\nEnter the maximum instances of each resource:\n");

	for(cntRes=0;cntRes<res;cntRes++)
	{
		printf("\n\tNo. of instances of resource #%d: ",cntRes);
		scanf("%d",&total[cntRes]);
	}

	 printf("\nEnter the maximum requirement of each process:\n");

	 for(cntProcess=0;cntProcess<process;cntProcess++)
	{
		printf("\nProcess #%d:\n",cntProcess);
```

```
        for(cntRes=0;cntRes<res;cntRes++)
                {
                        printf("\n\tRequirement for resource #%d: ",cntRes);
                        scanf("%d",&max[cntProcess][cntRes]);
                }
}

printf("\nEnter the current allocation for each process:\n");

for(cntProcess=0;cntProcess<process;cntProcess++)
{
        printf("\nProcess #%d:\n",cntProcess);
        for(cntRes=0;cntRes<res;cntRes++)
                {
                        printf("\n\tAllocation for resource #%d: ",cntRes);
                        scanf("%d",&allocated[cntProcess][cntRes]);
                }
}

/*Calculate available instances of each resource*/
for(cntRes=0;cntRes<res;cntRes++)
        {       //Sum of allocated instances
                for(cntProcess=0;cntProcess<process;cntProcess++)
                        {
                                available[cntRes]+=allocated[cntProcess][cntRes];
                        }
                //subtract from total no. of instances
                available[cntRes]=total[cntRes]-available[cntRes];
```

```
                }

        printf("\nThe available instances of each resource are:\n");
        for(cntRes=0;cntRes<res;cntRes++)
        {
                printf("\nResource #%d: %d",cntRes,available[cntRes]);
        }
        /*Calculate the need matrix*/

        for(cntProcess=0;cntProcess<process;cntProcess++)
        {
                for(cntRes=0;cntRes<res;cntRes++)
                        {
                                need[cntProcess][cntRes]=max[cntProcess][cntRes]-
allocated[cntProcess][cntRes];
                        }
        }

        printf("\n\nThe NEED matrix is:\n\n");

        for(cntProcess=0;cntProcess<process;cntProcess++)
        {
                 for(cntRes=0;cntRes<res;cntRes++)
                        {
                                 printf("\t%d",need[cntProcess][cntRes]);
                        }
                        printf("\n\n");
        }
```

```
printf("\nThe processes are executed in the foll. sequence:\n\n");
executed=0;	//Init no. of processes executed
do
{
	for(cntProcess=0,execFlag=0;cntProcess<process;cntProcess++)
	{
		flag=0; //Reset flag
		if(finish[cntProcess]!=0)
			continue;
		else
		{
			for(cntRes=0;cntRes<res;cntRes++)
			{
				if(need[cntProcess][cntRes]>available[cntRes])
				{	//Check for UNALLOWED condition
					flag=1;
					break;
				}
			}

			if(flag==0)
			{
				printf("\tP%d",cntProcess);
				finish[cntProcess]=1;
				for(cntRes=0;cntRes<res;cntRes++)
				{
					available[cntRes]+=allocated[cntProcess][cntRes];
				}
				execFlag=1;
```

```
                                        executed++;
                                }
                        }
                }

                if(execFlag==0)
                {       //Loop has executed for all processes, but none executed!
                        printf("\n\nThe system is in an UNSAFE state!");
                        break;
                }
        }while(executed<process);
        printf("\n");
}
```

/***OUTPUT**

```
student@student-OptiPlex-390:~$ cd 38
student@student-OptiPlex-390:~/38$ gcc bankers.c
student@student-OptiPlex-390:~/38$  ./a/out
bash: ./a/out: No such file or directory
student@student-OptiPlex-390:~/38$ ./a.out
Enter no. of processes: 5

Enter no. of resources: 4

Enter the maximum instances of each resource:

        No. of instances of resource #0: 10
```

No. of instances of resource #1: 5

No. of instances of resource #2: ^C

student@student-OptiPlex-390:~/38$

student@student-OptiPlex-390:~/38$ gcc bankers.c

student@student-OptiPlex-390:~/38$ ./a.out

Enter no. of processes: 5

Enter no. of resources: 3

Enter the maximum instances of each resource:

No. of instances of resource #0: 10

No. of instances of resource #1: 5

No. of instances of resource #2: 7

Enter the maximum requirement of each process:

Process #0:

Requirement for resource #0: 7

Requirement for resource #1: 5

Requirement for resource #2: 3

Process #1:

Requirement for resource #0: 3

Requirement for resource #1: 2

Requirement for resource #2: 2

Process #2:

Requirement for resource #0: 9

Requirement for resource #1: 0

Requirement for resource #2: 2

Process #3:

Requirement for resource #0: 2

Requirement for resource #1: 2

Requirement for resource #2: 2

Process #4:

Requirement for resource #0: 4

Requirement for resource #1: 3

Requirement for resource #2: 3

Enter the current allocation for each process:

Process #0:

Allocation for resource #0: 0

Allocation for resource #1: 1

Allocation for resource #2: 0

Process #1:

Allocation for resource #0: 2

Allocation for resource #1: 0

Allocation for resource #2: 0

Process #2:

Allocation for resource #0: 3

Allocation for resource #1: 0

Allocation for resource #2: 2

Process #3:

Allocation for resource #0: 2

Allocation for resource #1: 1

Allocation for resource #2: 1

Process #4:

Allocation for resource #0: 0

Allocation for resource #1: 0

Allocation for resource #2: 2

The available instances of each resource are:

Resource #0: 3

Resource #1: 3

Resource #2: 2

The NEED matrix is:

7 4 3

1 2 2

6 0 0

0 1 1

4 3 1

The processes are executed in the foll. sequence:

P1 P3 P4 P0 P2

*/