8. Design, develop and implement a C/C++/Java program to implement Banker's algorithm. Assume suitable input required to demonstrate the results.

```
#include<stdio.h>
#include<stdlib.h>
int p,r,i,j,max[10][10],alloc[10][10],avail[10],need[10][10];
int request[10][10],finish[10],work[10];
int safety_algo()
{
        int k,count=0,flag;
        for(i=0;i<p;i++)
        finish[i]=0;
        for(j=0;j<r;j++)
        work[j]=avail[j];
        while(count<p)
        {
                k=0;
                for(i=0;i<p;i++)
                {
                        flag=0;
                        for(j=0;j<r;j++)
                        if(finish[i]!=0||need[i][j]>work[j])
                        {
                                flag=1;
                                break;
                        }
                        if(flag!=1)
                        {
                                finish[i]=1;
                                for(j=0;j<r;j++)
                                work[j]=work[j]+alloc[i][j];
                                printf("p%d\t",i);
                                count++;
                                k++;
                        }
                }
```

```
                if(k==0)
                {
                        return 0;
                }
        }
        return 1;
}

void resource_request(int i)
{
        int j;
        for(j=0;j<r;j++)
        {
                if(request[i][j]>need[i][j])
                {
                        printf("Error::request more than demand\n");
                        return;
                }
        }
        for(j=0;j<r;j++)
        {
                if(request[i][j]>avail[j])
                {
                        printf("request more than available,process has to wait\n");
                        return;
                }
        }
        for(j=0;j<r;j++)
        {
                alloc[i][j]=alloc[i][j]+request[i][j];
                avail[j]=avail[j]-request[i][j];
                need[i][j]=need[i][j]-request[i][j];
        }
        if(safety_algo()==1)
        printf("\n SAFE::process %d can be allocated\n",i);
```

```
        else
        printf("UNSAFE::process shas towait\n");
}

int main()
{
        printf("enter the number of processes:");
        scanf("%d",&p);
        printf("\n enter the number of resources:");
        scanf("%d",&r);
        printf("\n enter the max matrix\n");
        for(i=0;i<p;i++)
        for(j=0;j<r;j++)
                scanf("%d",&max[i][j]);
        printf("\n enter the alloc matrix\n");
        for(i=0;i<p;i++)
        for(j=0;j<r;j++)
                scanf("%d",&alloc[i][j]);
        printf("\n enter the available resources\n");
        for(j=0;j<r;j++)
                scanf("%d",&avail[j]);
        for(i=0;i<p;i++)
        for(j=0;j<r;j++)
                need[i][j]=max[i][j]-alloc[i][j];
        printf("\tNEED\t\n");
        for(i=0;i<p;i++)
        {
                for(j=0;j<r;j++)
                        printf("%d\t",need[i][j]);
                printf("\n");
        }
        if(safety_algo()==1)
                printf("\n current state of system::SAFE\n");
        else
                printf("\n current state of system::UNSAFE\n");
```

```
        printf("enter the new request(processid)\n");
        scanf("%d",&i);
        printf("\n enter the request\n");
        for(j=0;j<r;j++)
                scanf("%d",&request[i][j]);
        resource_request(i);
        return 0;
}
```

Compilation step:

```
cc      8.c
./a.out
```

RUN1:

```
enter the number of processes:5
enter the number of resources:3
enter the max matrix
7       5       3
3       2       2
9       0       2
2       2       2
4       3       3
enter the alloc matrix
0       1       0
2       0       0
3       0       2
2       1       1
0       0       2
enter the available resources
3       3       2
NEED
7       4       3
1       2       2
6       0       0
0       1       1
4       3       1
P1      P3      P4      P0      P2
```

current state of system::SAFE

enter the new request(processid)

1

enter the request

1 0 2

SAFE::process 1 can be allocated

RUN2:

enter the number of processes:5

enter the number of resources:3

enter the max matrix

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

enter the alloc matrix

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

enter the available resources

3 3 2

NEED

7 4 3

1 2 2

6 0 0

0 1 1

4 3 1

P1 P3 P4 P0 P2

current state of system::SAFE

enter the new request(processid)

2

enter the request

1 0 2

Error::request more than demand