NAME: SHIVANI.M.V
CLASS: 6 - 'C'
USN: 1CR18CS152

PROGRAM: 08

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

CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int k=0, output[10], d=0, t=0, ins[5], i, avail[5],
   allocated[10][5], need[10][5], MAX[10][5];
 int pno, P[0], j, rz, count=0;
 printf("Enter the number of resources:");
 scanf("%d", &rz);
 printf("\n Enter the max instances of each resources\n");
 for(i=0; i<rz; i++)
    {
        available[i]=0;
        Scanf("%d", &ins[i]);
    }
 printf("\n Enter the number of processes:");
 scanf("%d", &pno);
 printf("\n Enter the allocation matrix\n");
  for(i=0; i<rz; i++)
      printf("\n");
```

```
for (i=0; i<pno; i++)
 {
     P[i] = i;
     printf(" P[%od] ", P[i]+1);
     for(j=0; j<rz; j++)
       {
        scanf ("%od", &allocated [i][j]) ;
        available[j] += allocated [i][j];
       }
   }
printf("\n Enter the MAX matrix \n");
for(i=0 ; i<rz; i++) {
     available [i] = ins [i] - available [i];
 }
printf("\n");
for(i=0; i<pno; i++) {
  printf(" P[%od] ", i+1);
  for (j=0; j<rz; j++)
      scanf(" %od", &MAX[i][j]);
  }
 printf("\n");
 A: d = -1;
 for (i=0 ; i<pno; i++) {
     count = 0;
     t = P[i];
```

```
for(j=0; j<rz; j++)
  {
    need[i][j] = MAX[i][j] - allocated[i][j];
   if (need[i][j] <= available[j])
              count++;
   }
  if (count == rz)
   {
     output[k++] = P[i];
     for(j=0; j<rz; j++)
        available[j] += allocated[i][j];
    }
  else
      P[++d] = P[i];
}
if(d != -1) {
      pno = d+1;
      goto A;
}
printf("\t");
for(i=0; i<k; i++)
   printf("P[%d]", output[i]+1);
   printf(")");
   getch();
}.
```

OUTPUT 1)

Enter the max instances of each resources

10
8
6

Enter the number of processes : 5

Enter the allocation matrix

P[1]	1	0	0
P[2]	2	0	1
P[3]	3	1	0
P[4]	1	1	1
P[5]	0	0	3

Enter the MAX matrix

P[1]	5	3	1
P[2]	3	1	1
P[3]	4	3	2
P[4]	9	5	4
P[5]	4	4	4

(P[2] P[3] P[5] P[1] P[4])

OUTPUT 2)

Enter the number of resources : 3

Enter the max instances of each resources

8
7
5

Enter the number of processes : 4

Enter the allocation matrix

P[1]	0	1	0
P[2]	2	0	2
P[3]	3	1	0
P[4]	0	0	1

Enter the MAX matrix

P[1]	4	4	4
P[2]	3	2	2
P[3]	6	5	4
P[4]	4	3	2

(P[2] P[3] P[4] P[1]).