Ex.No: 9 DEAD LOCK PREVENTION

Date:

Aim

To determine whether the process and their request for resources are in a deadlocked state.

Algorithm

1. Mark each process that has a row in the Allocation matrix of all zeros.
2. Initialize a temporary vectorW to equal the Available vector.
3. Find an indexi such that processi is currently unmarked and thei th row ofQ
4. is less than or equal to W . That is,Q ik … Wk, for 1 … k … m . If no such row is
5. found, terminate the algorithm.
5. If such a row is found, mark processi and add the corresponding row of the
6. allocation matrix to W . That is, setWk = Wk + Aik, for 1 … k … m . Return
7. to step 3.

Program

```
#include<stdio.h>
#include<conio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;
void input();
void show();
void cal();

main()
{
   int i,j;
   printf("Deadlock Detection Algo\n");
   input();
   show();
   cal();
   getch();
}

void input()
{
int i,j;
   printf("Enter the no of Processes\t");
   scanf("%d",&n);
   printf("Enter the no of resource instances\t");
   scanf("%d", &r);

   printf("Enter the Max Matrix\n");
   for(i=0; i<n; i++)
      for(j=0; j<r; j++)
         scanf("%d", &max[i][j]);

   printf("Enter the Allocation Matrix\n");
   for(i=0; i<n; i++)
      for(j=0; j<r; j++)
         scanf("%d", &alloc[i][j]);
   printf("Enter the available Resources\n");
   for(j=0;j<r;j++)
      scanf("%d",&avail[j]);
}

void show()
{
   int i, j;
   printf("Process\t Allocation\t Max\t Available\t");
   for(i=0; i<n; i++)
   {
      printf("\nP%d\t   ", i+1);
      for(j=0; j<r; j++)
      {
         printf("%d ", alloc[i][j]);
      }
      printf("\t");
      for(j=0; j<r; j++)
      {
         printf("%d ", max[i][j]);
      }
      printf("\t");
      if(I == 0)
      {
         for(j=0; j<r; j++)
            printf("%d ", avail[j]);
      }
   }
}

void cal()
{
   int finish[100], temp, need[100][100], flag=1, k, c1=0;
   int dead[100];
   int safe[100];
   int i, j;
   for(i=0; i<n; i++)
   {
      finish[i] = 0;
   }
```

```
/*find need matrix */
for(i=0; i<n; i++)
{
   for(j=0; j<r; j++)
   {
      need[i][j]= max[i][j] - alloc[i][j];
   }
}

while(flag)
{
   flag=0;
   for(i=0;i<n;i++)
   {
      int c=0;
      for(j=0;j<r;j++)
      {
         if((finish[i]==0) && (need[i][j] <= avail[j]))
         {
            c++;
            if(c == r)
            {
                 for(k=0; k<r; k++)
                 {
                      avail[k] += alloc[i][j];
                      finish[i]=1;
                      flag=1;
                 }
                 if(finish[i] == 1)
                 {
                      i=n;
                 }
            }
         }
      }
   }
}
J = 0;
Flag = 0;
for(i=0; i<n; i++)
{
   if(finish[i] == 0)
   {
      dead[j] = i;
      j++;
      flag = 1;
   }
}
```

```
   if(flag == 1)
   {
      printf("\n\nSystem  is  in  Deadlock  and  the  Deadlock
process are\n");
      for(i=0;i<n;i++)
      {
         printf("P%d\t", dead[i]);
      }
  }
  else
  {
      printf("\nNo Deadlock Occur");
  }
}
```

Output

```
********** Deadlock Detection Algo ************
Enter the no of Processes       3
Enter the no of resource instances      3
Enter the Max Matrix
3 6 0
4 3 3
3 4 4
Enter the Allocation Matrix
3 3 3
2 0 3
1 2 4
Enter the available Resources
1 2 0
Process  Allocation      Max     Available
P1          3 3 3        3 6 0   1 2 0
P2          2 0 3        4 3 3
P3          1 2 4        3 4 4

System is in Deadlock and the Deadlock process are
P0      P1      P2
```

Result

Thus using given state of information deadlocked process were determined.