Ex No: 3
Date: 13.02.2023

IMPLEMENTATION OF DEADLOCK AVOIDANCE ALGORITHM

Aim
To write a 'C' program by avoiding the deadlock using banker's algorithm.

Algorithm
1. Get the allocated and maximum resources and process name from the user.
2. Calculate the available matrix from the maximum and allocated resources.
3. Then calculate the need matrix.
4. Then apply the banker's algorithm to allocate the requested resource from available
5. resources.
6. Release the allocated resource from completed process.
7. This step is repeated to find the safe state among the available process.

Program
```
#include<stdio.h>
int main()
{
   int r,p;
   printf("enter number of resource:");
   scanf("%d",&r);
   printf("enter number of process:");
   scanf("%d",&p);
   int req[r],allo[p][r],res[r],max[p][r],ava[r],need[p][r];
   printf("enter instances of each resources:\n");
   for(int i=0;i<r;i++){
      scanf("%d",&res[i]);
   }
   printf("enter allocation of %d processes:\n",p);
   for(int I=0;I<p;I++){
      for(int j=0;j<r;j++){
         scanf("%d",&allo[I][j]);
      }
   }
   printf("enter max of %d process :\n",p);
   for(int I=0;I<p;I++){
      for(int j=0;j<r;j++){
         scanf("%d",&max[I][j]);
      }
   }
   printf("enter available of %d resources :\n",r);
   for(int I=0;I<r;I++){
      scanf("%d",&ava[I]);
   }
   for(int I=0;I<p;I++){
      for(int j=0;j<r;j++){
         need[I][j]=max[I][j]-allo[I][j];
```

```
    }
}

printf("\n\n---------- DEADLOCK AVOIDANCE ----------\n\n");

printf("allocation  max  need  available\n");
for(int I=0;I<p;I++){
    for(int j=0;j<r;j++){
        printf("%d ",allo[I][j]);
    }
    printf("  ");
    for(int j=0;j<r;j++){
        printf("%d ",max[I][j]);
    }
    printf("  ");
    for(int j=0;j<r;j++){
        printf("%d ",need[I][j]);
    }
    printf("  ");
    for(int j=0;j<r&&I==0;j++){
        printf("%d ",ava[j]);
    }
    printf("\n");
}
int visit[p],count=0,val=0,check=1,index=0;
char ch='A',str[p];
for(int I=0;I<p;I++){
    visit [I]=0;
}

while(check==1 && count<3*p){
    if(visit[index]==0){
        for(int I=0;I<r;I++){
            if(need[index][I]>ava[I])
            {
                index++;
                break;
        }
        }
        visit[index]=1;
        str[val++]=ch+index;
        for(int I=0;I<r;I++){
            ava[I]+=allo[index][I];
        }
        index++;
        if(index==p){
            index=0;
        }
    }
```

```
        else
        {
        index++;
        if(index==p){
              index=0;
           }
        }
        for(int I=0;I<p;I++){
           if(visit[I]==1){
              check=0;
           }
           if(visit[I]==0){
              check=1;
              break;
           }
        }
        count++;
     }

     int flag=0;
     for(int i=0;i<r;i++){
        if(ava[i]!=res[i]){
           flag=1;
           break;
        }
     }
     if(flag==0){
     printf("\n\nSafe Sequence:\n< ");
     for(int I=0;I<p;I++){
        printf("%c ",str[I]);
     }
     printf(" >");
     }
     else{
        printf("Unsafe Sequence");
     }
}
```

Output

```
"C:\Users\nanda\Desktop\jhansi\os\bankers algorithm.exe"
enter number of resource:3
enter number of process:5
enter instances of each resources:
10
5
7
enter allocation of 5 processes:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
enter max of 5 process :

7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
enter available of 3 resources :
3 3 2


-------------- DEADLOCK AVOIDANCE --------------

allocation    max         need    available
0 1 0         7 5 3      7 4 3     3 3 2
2 0 0         3 2 2      1 2 2
3 0 2         9 0 2      6 0 0
2 1 1         2 2 2      0 1 1
0 0 2         4 3 3      4 3 1


Safe Sequence:
< B D E A C  >
Process returned 0 (0x0)   execution time : 32.861 s
Press any key to continue.
```

Observation (20)	
Record (5)	
Total (25)	
Initial	

Result

Thus, C program for deadlock avoidance using banker's algorithm was written successfully and output is verified.