Ex. No. 9	DEADLOCK AVOIDANCE
Date: 28.03.2025	

Aim:

To find out a safe sequence using Banker's algorithm for deadlock avoidance.

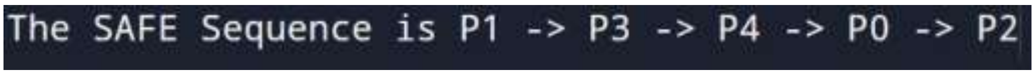
Program:

```
#include <stdio.h>
#define P 5
#define R 3

int main(){
   int i, j, k;
   int alloc[P][R] = { {0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}
};    int max[P][R] = { {7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}
};    int avail[R] = {3, 3, 2};    int f[P], ans[P], ind=0;
for(k=0;k<P;k++) f[k]=0;
   int need[P][R];    for(i=0;i<P;i++)
for(j=0;j<R;j++)
need[i][j]=max[i][j]-alloc[i][j];
   for(k=0;k<P;k++){
for(i=0;i<P;i++){
if(f[i]==0){              int flag=0;
for(j=0;j<R;j++){
if(need[i][j]>avail[j]){
                  flag=1;
break;
               }
            }
            if(flag==0){
for(j=0;j<R;j++)
avail[j]+=alloc[i][j];
ans[ind++]=i;
               f[i]=1;
            }
         }
      }
   }
```

```
    int flag=1;
for(i=0;i<P;i++){
if(f[i]==0){
flag=0;
            printf("The system is not in a safe state\n");
break;
        }
    }
    if(flag==1){        printf("The SAFE
Sequence is ");
        for(i=0;i<P-1;i++)
printf("P%d -> ",ans[i]);
printf("P%d\n",ans[P-1]);
    }
    return 0;
}
```

Output:



```
The SAFE Sequence is P1 -> P3 -> P4 -> P0 -> P2
```

Result:

The program to find the safe sequence using Banker’s Algorithm for deadlock avoidance was executed successfully.