This is the program for deadlock prevention using different processes and resurces allocated to them:

Question-11

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int Pr=4,Re=4,i,j,p,k;
        printf("Enter number of process and resources:  ");
        scanf("%d%d",&Pr,&Re);
        const int P=Pr,R=Re;
   int avail[15];
   printf("Enter number of available resources:");
        for(i=0;i<R;i++)
        {
                scanf("%d",&avail[i]);
        }
   int maxm[10][10];
   for(i=0;i<P;i++)
        {
                printf("Enter maximum required resources for P%d:",i+1);
                for(j=0;j<R;j++)
                {
                        scanf("%d",&maxm[i][j]);
                }
        }
   int allot[10][10];
   for(i=0;i<P;i++)
        {
                printf("Enter resources allocated by P%d:          ",i+1);
```

```
                for(j=0;j<R;j++)
                {
                        scanf("%d",&allot[i][j]);
                }
        }
int need[P][R];

 for (i = 0 ; i < P ; i++)
    for (j = 0 ; j < R ; j++)
        need[i][j] = maxm[i][j] - allot[i][j];
 bool finish[P] = {0};
 int safeSeq[P];
 int work[R];
 for (i = 0; i < R ; i++)
    work[i] = avail[i];
 int count = 0;
        while (count < P)
{
    bool found = false;
    for (p = 0; p < P; p++)
    {
        if (finish[p] == 0)
        {
            int j;
            for (j = 0; j < R; j++)
                if (need[p][j] > work[j])
                    break;
            if (j == R)
            {
```

```
                for (k = 0 ; k < R ; k++)
                    work[k] += allot[p][k];
                safeSeq[count++] = p;
                finish[p] = 1;

                found = true;
            }
          }
  }
      if (found == false)
      {
         printf("System is not in safe state");
      }
   }
   printf("System is in safe state\n");
   printf("Sequence is: ");
   for (int i = 0; i < P ; i++)
      printf("P%d_____",safeSeq[i]+1);

}
```