Program 7: Deadlock handling using Banker's algorithm:

```
#include <stdio.h>
#define MAX 16

int avail[MAX], max[MAX][MAX];
int allot[MAX][MAX];
int completed[MAX] = {0};

int need(int i, int j) {
      return max[i][j] - allot[i][j];
}

void read_matrix(int process, int resource, int matrix[][MAX]) {
      int i, j, instance;
      for (i = 0; i < process; i++) {
            printf("P[%d]: ", i);
            for (j = 0; j < resource; j++) {
                  scanf("%d", &instance);
                  matrix[i][j] = instance;
            }
      }
}

int main() {
      int process, resource, i, j, instance, k = 0;
      int count1 = 0, count2 = 0;
      printf("Num. of processes & resources: ");
      scanf("%d %d", &process, &resource);

      printf("Num. of available instances: ");
      for (i=0; i < resource; i++) {
            scanf("%d", &instance);
            avail[i] = instance;
      }
      printf("Enter Max. Requirement matrix:\n");
      read_matrix(process, resource, max);
      printf("Enter Allocation Matrix:\n");
      read_matrix(process, resource, allot);

      printf("Safe sequence:\n");
      while (count1 != process) {
            count2 = count1;
            for (i=0; i<process; i++) {
                  for (j = 0; j<resource; j++) {
                        if(need(i,j) <= avail[j]) {
                              k++;
                        }
                  }
                  if (k == resource && completed[i] == 0) {
                        printf("Complete: P[%d]\n", i);
                        completed[i] = 1;
                        for (j = 0; j < resource; j++) {
                              avail[j] += allot[i][j];
                        }
                        count1++;
                  }
                  k = 0;
            }
      }
}
```

Output:

```
mahesh@mahesh:~/Code/Lab/OS$ ./banker
Num. of processes & resources: 5 3
Num. of available instances: 3 3 2
Enter Max. Requirement matrix:
P[0]: 7 5 3
P[1]: 3 2 2
P[2]: 9 0 2
P[3]: 2 2 2
P[4]: 4 3 3
Enter Allocation Matrix:
P[0]: 0 1 0
P[1]: 2 0 0
P[2]: 3 0 2
P[3]: 2 1 1
P[4]: 0 0 2
Safe sequence:
Complete: P[1]
Complete: P[3]
Complete: P[4]
Complete: P[0]
Complete: P[2]
mahesh@mahesh:~/Code/Lab/OS$
```