

Write a C program to simulate Bankers algorithm for the purpose of deadlock avoidance.

INPUT:

```
1 #include <stdio.h>
2 #include <stdbool.h>
3 void calculateNeed(int need[][10], int max[][10], int allot[][10], int P, int R) {
4     for (int i = 0; i < P; i++)
5         for (int j = 0; j < R; j++)
6             need[i][j] = max[i][j] - allot[i][j];
7 }
8 bool isSafe(int processes[], int avail[], int max[][10], int allot[][10], int P, int R) {
9     int need[10][10];
10    calculateNeed(need, max, allot, P, R);
11    bool finish[10] = {0};
12    int safeSeq[10];
13    int work[10];
14    for (int i = 0; i < R; i++)
15        work[i] = avail[i];
16    int count = 0;
17    while (count < P) {
18        bool found = false;
19        for (int p = 0; p < P; p++) {
20            if (finish[p] == 0) {
21                int j;
22                for (j = 0; j < R; j++)
23                    if (need[p][j] > work[j])
24                        break;
25                if (j == R) {
26                    for (int k = 0; k < R; k++)
27                        work[k] += allot[p][k];
28                    safeSeq[count++] = p;
29                    finish[p] = 1;
30                    found = true;
31                }
32            }
33        }
34        if (!found) {
35            printf("System is not in safe state\n");
36            return false;
37        }
38        printf("SYSTEM IS IN SAFE STATE\n");
39        printf("The Safe Sequence is -- ");
40        for (int i = 0; i < P; i++)
41            printf("%d ", safeSeq[i]);
42        printf("\n");
43        return true;
44    }
45 }
46 bool requestResource(int processes[], int avail[], int max[][10], int allot[][10], int P, int R, int pid, int request[]) {
```

```

50 bool requestResource(int processes[], int avail[], int max[][10], int allot[][10], int P, int R, int pid, int request[]) {
51     int need[10][10];
52     calculateNeed(need, max, allot, P, R);
53     for (int i = 0; i < R; i++) {
54         if (request[i] > need[pid][i]) {
55             printf("Error: Process has exceeded its maximum claim\n");
56             return false;
57         }
58     }
59     for (int i = 0; i < R; i++) {
60         if (request[i] > avail[i]) {
61             printf("Resources are not available, process must wait\n");
62             return false;
63         }
64     }
65
66     for (int i = 0; i < R; i++) {
67         avail[i] -= request[i];
68         allot[pid][i] += request[i];
69         need[pid][i] -= request[i];
70     }
71     if (isSafe(processes, avail, max, allot, P, R) == false) {
72         for (int i = 0; i < R; i++) {
73             avail[i] += request[i];
74             allot[pid][i] -= request[i];
75             need[pid][i] += request[i];
76         }
77         printf("System is not in safe state after allocation\n");
78         return false;
79     }
80     printf("Resources have been allocated successfully\n");
81     return true;
82 }
83
84 int main() {
85     int P, R;
86     printf("Enter number of processes: ");
87     scanf("%d", &P);
88     printf("Enter number of resources: ");
89     scanf("%d", &R);
90
91     int processes[10];
92     for (int i = 0; i < P; i++) {
93         processes[i] = 1;
94     }
95     int avail[10];
96     int max[10][10];
97     int allot[10][10];
98     for (int i = 0; i < P; i++) {
99         printf("Enter allocation for P%d: ", i);
100         for (int j = 0; j < R; j++) {
101             processes[i] = i;
102             int avail[10];
103             int max[10][10];
104             int allot[10][10];
105             for (int i = 0; i < P; i++) {
106                 printf("Enter allocation for P%d: ", i);
107                 for (int j = 0; j < R; j++) {
108                     scanf("%d", &allot[i][j]);
109                 }
110                 printf("Enter max for P%d: ", i);
111                 for (int j = 0; j < R; j++) {
112                     scanf("%d", &max[i][j]);
113                 }
114             }
115             printf("Enter available resources: ");
116             for (int i = 0; i < R; i++) {
117                 scanf("%d", &avail[i]);
118             }
119             int need[10][10];
120             calculateNeed(need, max, allot, P, R);
121             printf("\nProcess\tAllocation\tMax\t\tNeed\n");
122             for (int i = 0; i < P; i++) {
123                 printf("P%d\t", i);
124                 for (int j = 0; j < R; j++) {
125                     printf("%d ", allot[i][j]);
126                 }
127                 printf("\t\t");
128                 for (int j = 0; j < R; j++) {
129                     printf("%d ", max[i][j]);
130                 }
131                 printf("\t\t");
132                 for (int j = 0; j < R; j++) {
133                     printf("%d ", need[i][j]);
134                 }
135                 printf("\n");
136             }
137             isSafe(processes, avail, max, allot, P, R);
138             int pid, request[10];
139             printf("Enter PID for new request: ");
140             scanf("%d", &pid);
141             printf("Enter request for resources: ");
142             for (int i = 0; i < R; i++) {
143                 scanf("%d", &request[i]);
144             }
145             requestResource(processes, avail, max, allot, P, R, pid, request);
146         }
147     }
148     return 0;
149 }

```

OUTPUT:

```
Enter number of processes: 5
Enter number of resources: 3
Enter allocation for P0: 0
1
0
Enter max for P0: 7
5
3
Enter allocation for P1: 2
0
0
Enter max for P1: 3
2
2
Enter allocation for P2: 3
0
2
Enter max for P2: 9
0
2
Enter allocation for P3: 2
1
1
Enter max for P3: 2
2
2
Enter allocation for P4: 0
0
2
Enter max for P4: 4
3
3
Enter available resources: 3
3
2

Process Allocation      Max      Need
P0      0 1 0      7 5 3      7 4 3
P1      2 0 0      3 2 2      1 2 2
P2      3 0 2      9 0 2      6 0 0
P3      2 1 1      2 2 2      0 1 1
P4      0 0 2      4 3 3      4 3 1

SYSTEM IS IN SAFE STATE
The Safe Sequence is -- P1 P3 P4 P0 P2
Enter PID for new request: 1
Enter request for resources: 1
0
2
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- P1 P3 P4 P0 P2
Resources have been allocated successfully
```