

## **LAB 08**

**Course: CT-353-Operating Systems**

**Department: BCIT (Specialisation in Data Science)**

**Instructor's Name: Muhammed Abdullah Siddiqui**

**Student Name: Maryam Ashraff (DT-22050)**



```

1  #include <stdio.h>
2  #include <conio.h>
3
4  int max[100][100];
5  int alloc[100][100];
6  int need[100][100];
7  int avail[100];
8  int n, r;
9
10 void input();
11 void show();
12 void cal();
13
14 int main() {
15     printf("***** Deadlock Detection Algorithm *****\n");
16     input();
17     show();
18     cal();
19     getch();
20     return 0;
21 }
22
23 void input() {
24     int i, j;
25     printf("Enter the number of Processes:\t");
26     scanf("%d", &n);
27     printf("Enter the number of Resource Instances:\t");
28     scanf("%d", &r);
29
30     printf("Enter the Max Matrix:\n");
31     for (i = 0; i < n; i++) {
32         for (j = 0; j < r; j++) {
33             scanf("%d", &max[i][j]);
34         }
35     }
36
37     printf("Enter the Allocation Matrix:\n");
38     for (i = 0; i < n; i++) {
39         for (j = 0; j < r; j++) {
40             scanf("%d", &alloc[i][j]);
41         }
42     }
43
44     printf("Enter the Available Resources:\n");

```

```

49
50 void show() {
51     int i, j;
52     printf("\nProcess\tAllocation\tMax\t\tAvailable\n");
53     for (i = 0; i < n; i++) {
54         printf("P%d\t", i + 1);
55         for (j = 0; j < r; j++) {
56             printf("%d ", alloc[i][j]);
57         }
58
59         printf("\t");
60         for (j = 0; j < r; j++) {
61             printf("%d ", max[i][j]);
62         }
63
64         printf("\t");
65         if (i == 0) {
66             for (j = 0; j < r; j++) {
67                 printf("%d ", avail[j]);
68             }
69         }
70         printf("\n");
71     }
72 }
73
74 void cal() {
75     int finish[100], dead[100];
76     int i, j, k, c, c1 = 0, flag = 1;
77
78     // Initialize finish array
79     for (i = 0; i < n; i++) {
80         finish[i] = 0;
81     }
82
83     // Calculate need matrix
84     for (i = 0; i < n; i++) {
85         for (j = 0; j < r; j++) {
86             need[i][j] = max[i][j] - alloc[i][j];
87         }
88     }
89
90     while (flag) {
91         flag = 0;
92         for (i = 0; i < n; i++) {

```

```

87     }
88 }
89
90 while (flag) {
91     flag = 0;
92     for (i = 0; i < n; i++) {
93         int can_execute = 1;
94         if (finish[i] == 0) {
95             for (j = 0; j < r; j++) {
96                 if (need[i][j] > avail[j]) {
97                     can_execute = 0;
98                     break;
99                 }
100             }
101             if (can_execute) {
102                 for (k = 0; k < r; k++) {
103                     avail[k] += alloc[i][k];
104                 }
105                 finish[i] = 1;
106                 flag = 1;
107             }
108         }
109     }
110 }
111
112 int deadlock_found = 0, dead_count = 0;
113 for (i = 0; i < n; i++) {
114     if (finish[i] == 0) {
115         dead[dead_count++] = i;
116         deadlock_found = 1;
117     }
118 }
119
120 if (deadlock_found) {
121     printf("\n\nSystem is in Deadlock.\nDeadlocked Processes are:\n");
122     for (i = 0; i < dead_count; i++) {
123         printf("%d\t", dead[i] + 1);
124     }
125     printf("\n");
126 } else {
127     printf("\n\nNo Deadlock Detected. System is in a Safe State.\n");
128 }
129 }

```

```
C:\Users\marya\Downloads\O × + v
***** Deadlock Detection Algorithm *****
Enter the number of Processes: 5
Enter the number of Resource Instances: 3
Enter the Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the Available Resources:
3 3 2

Process Allocation      Max      Available
P1      0 1 0    7 5 3    3 3 2
P2      2 0 0    3 2 2
P3      3 0 2    9 0 2
P4      2 1 1    2 2 2
P5      0 0 2    4 3 3

No Deadlock Detected. System is in a Safe State.
```

```
C:\Users\marya\Downloads\O × + v
***** Deadlock Detection Algorithm *****
Enter the number of Processes: 4
Enter the number of Resource Instances: 3
Enter the Max Matrix:
3 2 2
6 1 3
3 1 4
4 2 2
Enter the Allocation Matrix:
1 0 0
5 1 1
2 1 1
0 0 2
Enter the Available Resources:
0 0 0

Process Allocation      Max      Available
P1      1 0 0    3 2 2    0 0 0
P2      5 1 1    6 1 3
P3      2 1 1    3 1 4
P4      0 0 2    4 2 2

System is in Deadlock.
Deadlocked Processes are:
P1      P2      P3      P4
|
```