ASHNA SHAIKH [DT-22019]

OPERATING SYSTEM LAB [CT-353]

LAB 08

CODE:

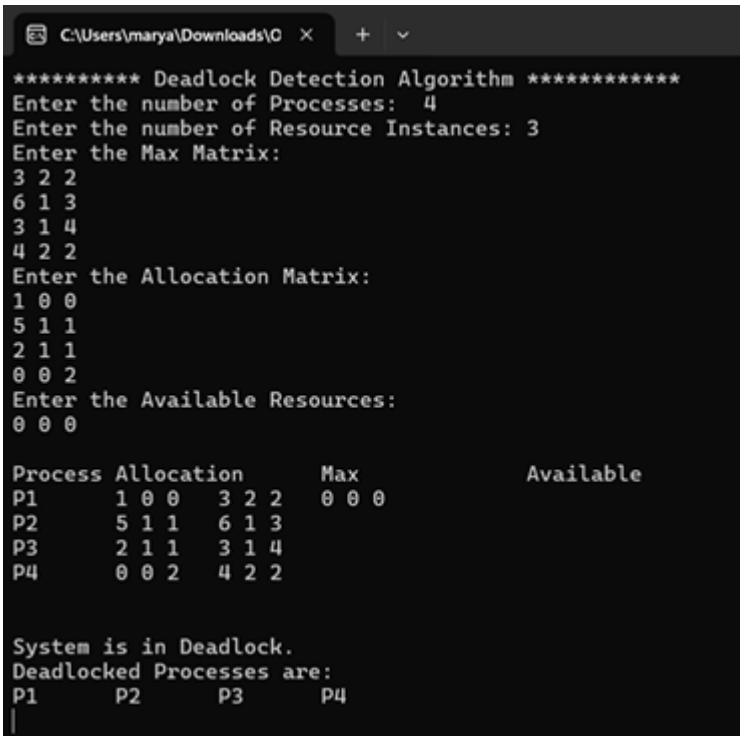
```c
#include <stdio.h>
#include <conio.h>

int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, r;

void input();
void show();
void cal();

int main() {
    printf("********** Deadlock Detection Algorithm ************\n");
    input();
    show();
    cal();
    getch();
    return 0;
}

void input() {
    int i, j;
    printf("Enter the number of Processes:\t");
    scanf("%d", &n);
    printf("Enter the number of Resource Instances:\t");
    scanf("%d", &r);

    printf("Enter the Max Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < r; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("Enter the Allocation Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < r; j++) {
            scanf("%d", &alloc[i][j]);
        }
    }

    printf("Enter the Available Resources:\n");
```

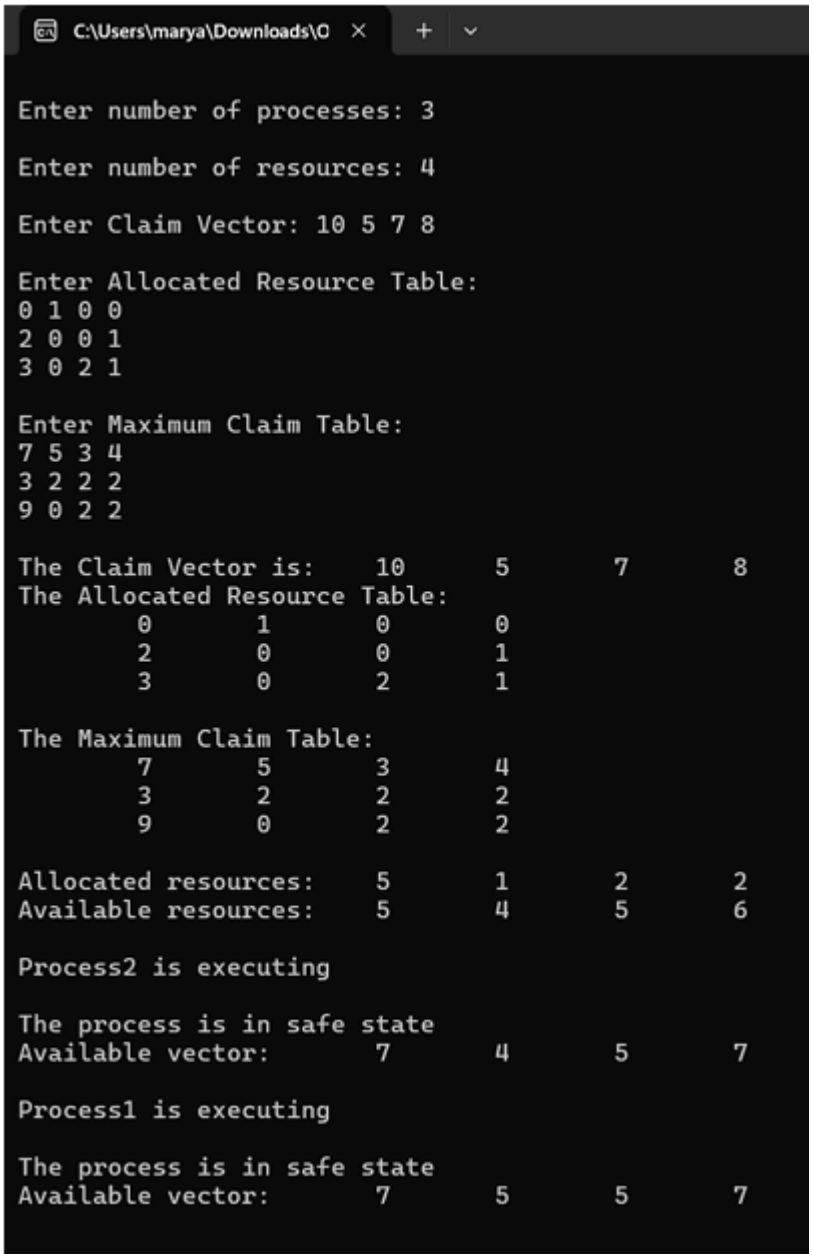
```
49
50  void show() {
51      int i, j;
52      printf("\nProcess\tAllocation\tMax\t\tAvailable\n");
53      for (i = 0; i < n; i++) {
54          printf("P%d\t", i + 1);
55          for (j = 0; j < r; j++) {
56              printf("%d ", alloc[i][j]);
57          }
58
59          printf("\t");
60          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++) {
```

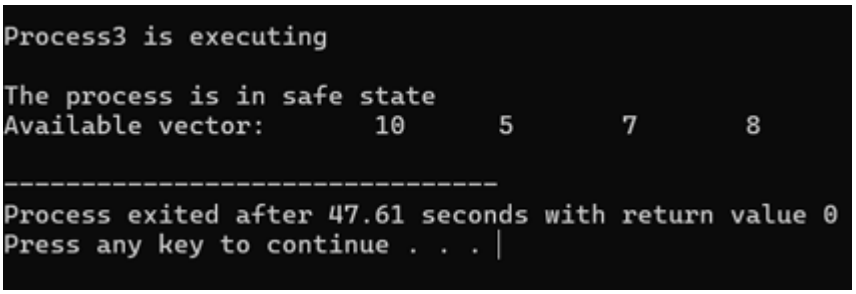
```
    }
}

while (flag) {
    flag = 0;
    for (i = 0; i < n; i++) {
        int can_execute = 1;
        if (finish[i] == 0) {
            for (j = 0; j < r; j++) {
                if (need[i][j] > avail[j]) {
                    can_execute = 0;
                    break;
                }
            }
            if (can_execute) {
                for (k = 0; k < r; k++) {
                    avail[k] += alloc[i][k];
                }
                finish[i] = 1;
                flag = 1;
            }
        }
    }
}

int deadlock_found = 0, dead_count = 0;
for (i = 0; i < n; i++) {
    if (finish[i] == 0) {
        dead[dead_count++] = i;
        deadlock_found = 1;
    }
}

if (deadlock_found) {
    printf("\n\nSystem is in Deadlock.\nDeadlocked Processes are:\n");
    for (i = 0; i < dead_count; i++) {
        printf("P%d\t", dead[i] + 1);
    }
    printf("\n");
} else {
    printf("\n\nNo Deadlock Detected. System is in a Safe State.\n");
}
}
```

OUTPUT:


C:\Users\marya\Downloads\C
********** 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\C
********** 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


C:\Users\marya\Downloads\O
Enter number of processes: 3
Enter number of resources: 4
Enter Claim Vector: 10 5 7 8
Enter Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
Enter Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
The Claim Vector is: 10 5 7 8
The Allocated Resource Table:
0 1 0 0
2 0 0 1
3 0 2 1
The Maximum Claim Table:
7 5 3 4
3 2 2 2
9 0 2 2
Allocated resources: 5 1 2 2
Available resources: 5 4 5 6
Process2 is executing
The process is in safe state
Available vector: 7 4 5 7
Process1 is executing
The process is in safe state
Available vector: 7 5 5 7


Process3 is executing
The process is in safe state
Available vector: 10 5 7 8
Process exited after 47.61 seconds with return value 0
Press any key to continue . . .