

Naming the file: **20BCE1798\_ANSHGOEL\_Ex6\_BANKER'S ALGORITHM**

Reg No.: **20BCE1798**

Name: **ANSH GOEL**

Course Code: CSE2005

Course Name: Operating Systems (Embedded Lab)

Slot: L27+L28

Ex No 6:

Date: 25-02-22

**Ex. No 6**

**Date: 25-02-22**

### **Banker's Algorithm**

A restaurant has 5 tables. Assume that there are initially 10 numbers of plates, 9 numbers of spoons and 5 numbers of forks. 5 groups of customers come and occupy 5 tables. Each group are requesting for some amount of plates out of which only few were provided to them due to minimum number of available resources. Table 1 is initially provided with 2, 3, 2 out of 6, 5, 4 number of plates, spoons and forks respectively. Table 2 is initially provided with 2, 2, 0 out of 4, 2, 0 number of plates, spoons and forks respectively. Table 3 is initially provided with 1, 1, 1 out of 3, 1, 2 number of plates, spoons and forks respectively. Table 4 is initially provided with 0, 0, 0 out of 2, 2, 2 number of plates, spoons and forks respectively. Table 5 is initially provided with 2, 3, 2 out of 5, 5, 3 number of plates, spoons and forks respectively.

- i. Identify the further required resources in each table.
- ii. Could the requirement of all tables satisfy? If so, identify the sequence in which the tables are provided with the required resources, if any
- iii. If Table 5 requests for 2 plates immediately out of its further requirement, is it possible to provide the same? If so, identify the further required resources in each table after satisfying the request.
- iv. Could the requirement of all tables satisfy after satisfying the request? If so, identify the sequence in which the tables are provided with the required resources, if any

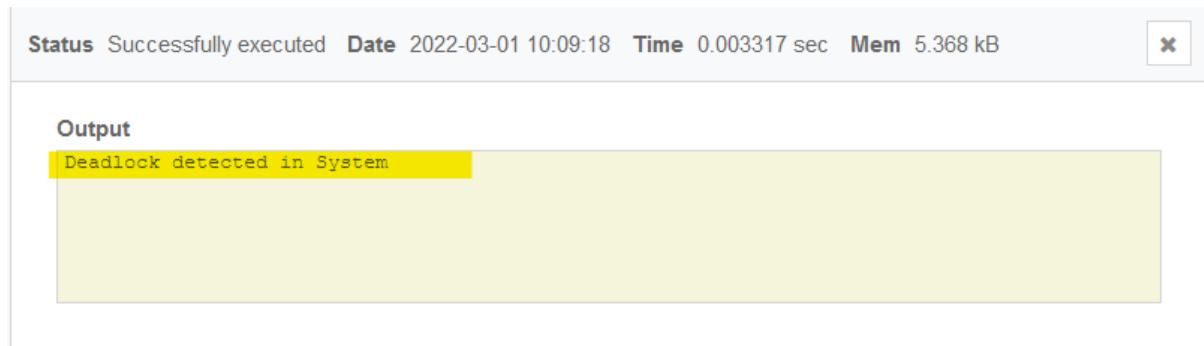
Write a C program for the above scenario.

Note: Kindly follow the upload format

## CODE:

```
main.cpp
1 #include <stdio.h>
2 #include<stdlib.h>
3 #include<stdbool.h>
4 int
5 main ()
6 {
7     int n = 5, m = 3, i = 0, j = 0, k = 0;
8     int total[] = { 10, 9, 5 };
9     int alloc[5][3] = {
10         {2, 3, 2}, {2, 2, 0}, {1, 1, 1}, {0, 0, 0}, {2, 3, 2}
11     };
12     int max[5][3] = { {6, 5, 4}, {4, 2, 0}, {3, 1, 2}, {2, 2, 2}, {5, 5, 3} };
13     int ava[m], ans[n], ind = 0;
14     //int ava[3]={3,0,1};
15     int need[n][m];
16     for (i = 0; i < n; i++)
17         for (j = 0; j < m; j++)
18             need[i][j] = max[i][j] - alloc[i][j];
19     for (i = 0; i < m; i++)
20         for (j = 0; j < n; j++)
21             total[i] -= alloc[j][i];
22     for (i = 0; i < m; i++)
23         ava[i] = total[i];
24     bool finish[n];
25     int count = 0;
26     while (count < n)
27     {
28         bool found = false;
29         for (int i = 0; i < n; i++)
30         {
31             if (finish[i] == 0)
32             {
33                 for (j = 0; j < m; j++)
34                     if (need[i][j] > ava[j])
35                         break;
36                     if (j == m)
37                     {
38                         for (int k = 0; k < m; k++)
39                             ava[k] += alloc[i][k];
40                         ans[count++] = i;
41                         finish[i] = 1;
42                         found = true;
43                     }
44                 }
45             }
46             if (found == false)
47             {
48                 printf ("Deadlock detected in System");
49                 return false;
50             }
51         }
52         printf ("System is in safe state\n");
53         printf ("Safe sequence is:\n");
54         for (int i = 0; i < n; i++)
55             printf ("Table %d ", ans[i] + 1);
56     }
57 }
```

**OUTPUT:**



The screenshot shows a software interface with a header bar containing status information: "Status Successfully executed Date 2022-03-01 10:09:18 Time 0.003317 sec Mem 5.368 kB". Below the header is a section titled "Output" which contains the text "Deadlock detected in System".

**i.**

For the system to be in safe state, at least 1 more fork is required. This one extra fork should be given to Table 3

**CODE:**

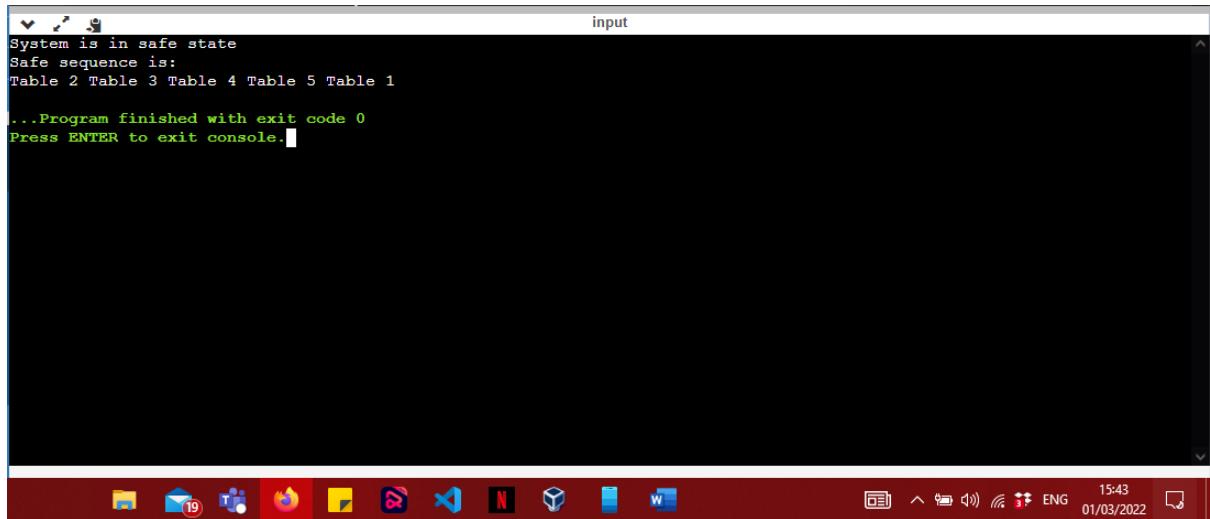
```

main.cpp

1 #include <stdio.h>
2 #include<stdlib.h>
3 #include<stdbool.h>
4 int main()
5 {
6     int n=5,m=3,i=0,j=0,k=0;
7     int total[]={10,9,6};
8     int alloc[5][3]={
9         {2,3,2},{2,2,0},{1,1,2},{0,0,0},{2,3,2}
10    };//Table 3 has been allocated one more fork
11    int max[5][3]={{6,5,4},{4,2,0},{3,1,2},{2,2,2},{5,5,3}};
12    int ava[m],ans[n],ind=0;
13    //int ava[3]={3,0,1};
14    int need[n][m];
15    for(i=0;i<n;i++)
16        for(j=0;j<m;j++)
17            need[i][j]=max[i][j]-alloc[i][j];
18    for(i=0;i<m;i++)
19        for(j=0;j<n;j++)
20            total[i]-=alloc[j][i];
21    for(i=0;i<m;i++)
22        ava[i]=total[i];
23    bool finish[n];
24    int count = 0;
25    while (count < n)
26    {
27        bool found = false;
28        for (int i = 0; i < n; i++)
29        {
30            if (finish[i] == 0)
31            {
32                for (j = 0; j < m; j++)
33                    if (need[i][j] > ava[j])
34                        break;
35                if (j == m)
36                {
37                    for (int k = 0 ; k < m ; k++)
38                        ava[k] += alloc[i][k];
39                    ans[count++] = i;
40                    finish[i] = 1;
41                    found = true;
42                }
43            }
44        }
45        if (found == false)
46        {
47            printf("Deadlock detected in System");
48            return false;
49        }
50    }
51    printf("System is in safe state\n");
52    printf("Safe sequence is:\n");
53    for (int i = 0; i < n ; i++)
54        printf("Table %d ",ans[i]+1);
55 }

```

## OUTPUT:



```
System is in safe state
Safe sequence is:
Table 2 Table 3 Table 4 Table 5 Table 1
...Program finished with exit code 0
Press ENTER to exit console.
```

ii.

If one more fork is provided to Table 3, then requirements of all tables can be satisfied.

The sequence in which the tables are provided with required resources is:

<Table2 Table3 Table4 Table5 Table1>

iii.

It is possible to satisfy the needs of Table 5 under the provided resources which 10 plates, 9 spoons and 6 forks where the extra fork is provided to Table 3. There is no requirement of more resources to satisfy the need of table 5 for 7 plates.

CODE:

```
main.cpp
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<stdbool.h>
4 int main()
5 {
6     int n=5,m=3,i=0,j=0,k=0;
7     int total[]={10,9,6};
8     int alloc[5][3]={{
9         {2,3,2},{2,2,0},{1,1,2},{0,0,0},{2,3,2}
10    }};
11    int max[5][3]={{6,5,4},{4,2,0},{3,1,2},{2,2,2},{7,5,3}};//Number of required of plates of Table 5 has been increased by 2
12    int ava[m],ans[n],ind=0;
13    //int ava[3]={3,0,1};
14    int need[n][m];
15    for(i=0;i<n;i++)
16        for(j=0;j<m;j++)
17            need[i][j]=max[i][j]-alloc[i][j];
18    for(i=0;i<m;i++)
19        for(j=0;j<n;j++)
20            total[i]=-alloc[j][i];
21    for(i=0;i<m;i++)
22        ava[i]=total[i];
23    bool finish[n];
24    int count = 0;
25    while (count < n)
26    {
27        bool found = false;
28        for (int i = 0; i < n; i++)
29        {
30            if (finish[i] == 0)
31            {
32                for (j = 0; j < m; j++)
33                    if (need[i][j] > ava[j])
34                        break;
35                if (j == m)
36                {
37                    for (int k = 0 ; k < m ; k++)
38                        ava[k] += alloc[i][k];
39                    ans[count++] = i;
40                    finish[i] = 1;
41                    found = true;
42                }
43            }
44        }
45        if (found == false)
46        {
47            printf("Deadlock detected in System");
48            return false;
49        }
50    }
51    printf("System is in safe state\n");
52    printf("Safe sequence is:\n");
53    for (int i = 0; i < n ; i++)
54        printf("Table %d ",ans[i]+1);
55    }
56
57 }
```

## OUTPUT:

```
input
System is in safe state
Safe sequence is:
Table 2 Table 3 Table 4 Table 5 Table 1
...Program finished with exit code 0
Press ENTER to exit console.
```

**iv.**

Yes, requirements of all tables are satisfied when Table 3 is provided one more fork. The sequence (Safety Sequence) in which the tables are provided required resources is:  
<Table 2 Table 3 Table 4 Table 5 Table 1>