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.

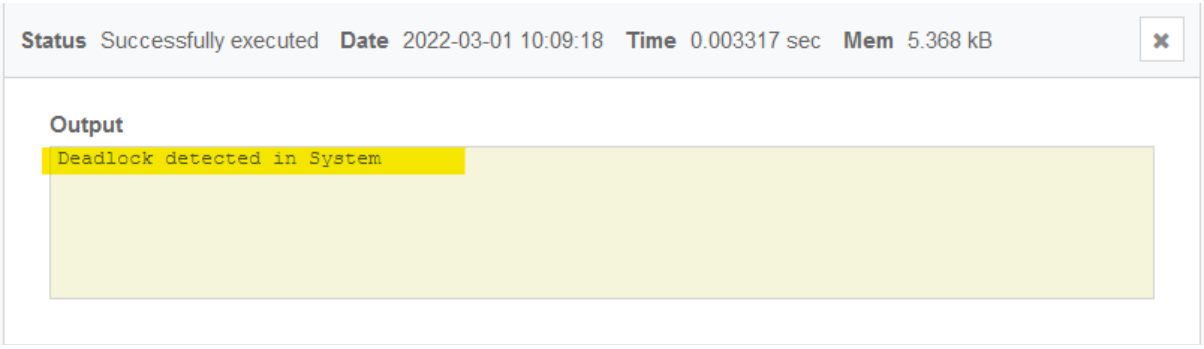
Note: Kindly follow the upload format

CODE:

main.cpp

```cpp
#include <stdio.h>
#include<stdlib.h>
#include<stdbool.h>
int
main ()
{
  int n = 5, m = 3, i = 0, j = 0, k = 0;
  int total[] = { 10, 9, 5 };
  int alloc[5][3] = {
    {2, 3, 2}, {2, 2, 0}, {1, 1, 1}, {0, 0, 0}, {2, 3, 2}
  };
  int max[5][3] = { {6, 5, 4}, {4, 2, 0}, {3, 1, 2}, {2, 2, 2}, {5, 5, 3} };
  int ava[m], ans[n], ind = 0;
  //int ava[3]={3,0,1};
  int need[n][m];
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      need[i][j] = max[i][j] - alloc[i][j];
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      total[i] -= alloc[j][i];
  for (i = 0; i < m; i++)
    ava[i] = total[i];
  bool finish[n];
  int count = 0;
  while (count < n)
    {
      bool found = false;
      for (int i = 0; i < n; i++)
      {
        if (finish[i] == 0)
          {
            for (j = 0; j < m; j++)
          if (need[i][j] > ava[j])
            break;
            if (j == m)
          {
            for (int k = 0; k < m; k++)
              ava[k] += alloc[i][k];
            ans[count++] = i;
            finish[i] = 1;
            found = true;
          }
          }
      }
        if (found == false)
      {
        printf ("Deadlock detected in System");
        return false;
      }
      }
    printf ("System is in safe state\n");
    printf ("Safe sequence is:\n");
    for (int i = 0; i < n; i++)
      printf ("Table %d ", ans[i] + 1);
}
```

OUTPUT:



i.

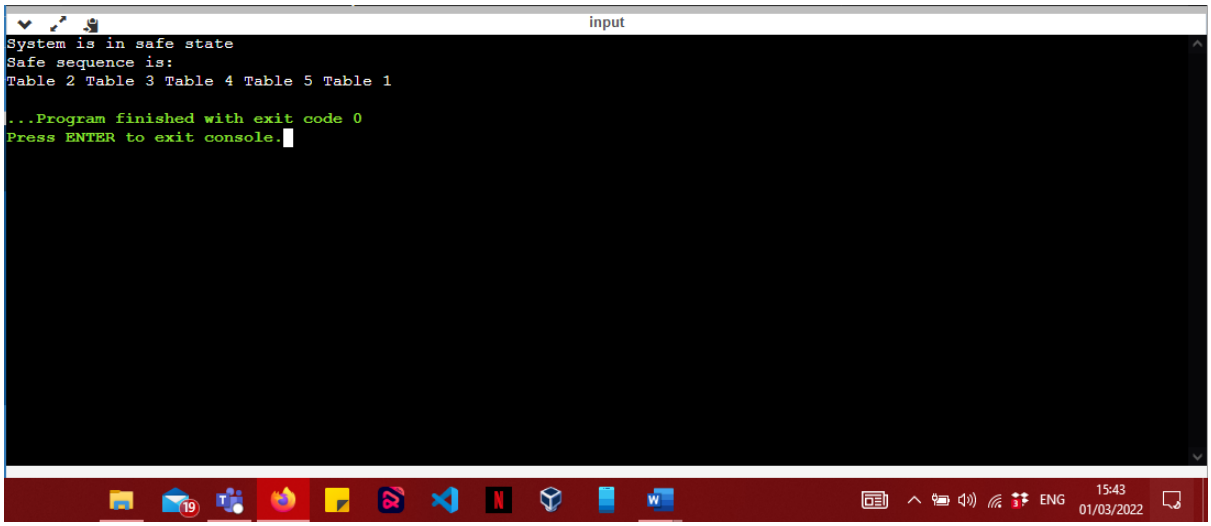
For the system to be in safe state, at least 1 more fork is required. This one extra fork should be given to Table 3

CODE:

main.cpp

```c
#include <stdio.h>
#include<stdlib.h>
#include<stdbool.h>
int main()
{
    int n=5,m=3,i=0,j=0,k=0;
    int total[]={10,9,6};
    int alloc[5][3]={
        {2,3,2},{2,2,0},{1,1,2},{0,0,0},{2,3,2}
    };//Table 3 has been allocated one more fork
    int max[5][3]={{6,5,4},{4,2,0},{3,1,2},{2,2,2},{5,5,3}};
    int ava[m],ans[n],ind=0;
    //int ava[3]={3,0,1};
    int need[n][m];
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    need[i][j]=max[i][j]-alloc[i][j];
    for(i=0;i<m;i++)
    for(j=0;j<n;j++)
    total[i]-=alloc[j][i];
    for(i=0;i<m;i++)
    ava[i]=total[i];
    bool finish[n];
    int count = 0;
    while (count < n)
    {
        bool found = false;
        for (int i = 0; i < n; i++)
        {
            if (finish[i] == 0)
            {
                for (j = 0; j < m; j++)
                    if (need[i][j] > ava[j])
                        break;
                if (j == m)
                {
                    for (int k = 0 ; k < m ; k++)
                    ava[k] += alloc[i][k];
                    ans[count++] = i;
                    finish[i] = 1;
                    found = true;
                }
            }
        }
        if (found == false)
        {
            printf("Deadlock detected in System");
            return false;
        }
    }
    printf("System is in safe state\n");
    printf("Safe sequence is:\n");
    for (int i = 0; i < n ; i++)
    printf("Table %d ",ans[i]+1);
}
```

OUTPUT:



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

```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
int main()
{
    int n=5,m=3,i=0,j=0,k=0;
    int total[]={10,9,6};
    int alloc[5][3]={
        {2,3,2},{2,2,0},{1,1,2},{0,0,0},{2,3,2}
    };
    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
    int ava[m],ans[n],ind=0;
    //int ava[3]={3,0,1};
    int need[n][m];
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    need[i][j]=max[i][j]-alloc[i][j];
    for(i=0;i<m;i++)
    for(j=0;j<n;j++)
    total[i]-=alloc[j][i];
    for(i=0;i<m;i++)
    ava[i]=total[i];
    bool finish[n];
    int count = 0;
    while (count < n)
    {
        bool found = false;
        for (int i = 0; i < n; i++)
        {
            if (finish[i] == 0)
            {
                for (j = 0; j < m; j++)
                    if (need[i][j] > ava[j])
                        break;
                if (j == m)
                {
                    for (int k = 0 ; k < m ; k++)
                    ava[k] += alloc[i][k];
                    ans[count++] = i;
                    finish[i] = 1;
                    found = true;
                }
            }
        }
        if (found == false)
        {
            printf("Deadlock detected in System");
            return false;
        }
    }
    printf("System is in safe state\n");
    printf("Safe sequence is:\n");
    for (int i = 0; i < n ; i++)
    printf("Table %d ",ans[i]+1);
}
```

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>