```cpp
//OS Lab Assignment: Deadlock Avoidance using Banker's Algorithm
//Submitted by: Sanskar Sharma
//Roll number: 90
//PRN: 0120180381


#include<iostream>
#include<string.h>

using namespace std;

int all_process_completed(int n,int finish[])
{
    int flag=1;
    for(int i=0;i<n;++i)
    {
        if(finish[i]==0)
        {
            flag=0;
            break;
        }
    }
    return flag;
}

int main()
{
    cout<<"\n\t\t\t\t    OS Lab Assignment";
    cout<<"\n\t\t\t\tSubmitted by: Sanskar Sharma";
    cout<<"\n\t\t\t\t     PRN: 0120180381"<<"\n";
    cout<<"\n\t\t\tDeadlock Avoidance using Banker's Algorithm";
    int n, m; //n: no of processes, m: no. of resources

    cout<<"\n\t\tEnter the number of processes: ";
    cin>>n;

    //name the procrsses
    cout<<"\n\t\tEnter the name of each process: ";
    char process_name[n][50];
    for(int i=0;i<n;++i)
    {
        cout<<"\n\t\t    Enter the name of "<<i+1<<" process: ";
        cin>>process_name[i];
    }


    cout<<"\n\t\tEnter the number of resources: ";
    cin>>m;

    //name the resouces
    cout<<"\n\t\tEnter the name of each resource: ";
    char resource_name[m][50];
    for(int i=0;i<m;++i)
    {
        cout<<"\n\t\t    Enter the name of "<<i+1<<" resource: ";
        cin>>resource_name[i];
    }

    //User input
    int allocation_matrix[n][m], max_matrix[n][m], available_array[m];

    //Input Allocation matrix
    cout<<"\n\t\tEnter the allocation matrix: ";
    for(int i=0;i<n;++i)
    {
        for(int j=0;j<m;++j)
        {
            cout<<"\n\t\t  Enter the allocated instances for process "<<process_name[i];
            cout<<" of resource "<<resource_name[j]<<" : ";
            cin>>allocation_matrix[i][j];
            while(allocation_matrix[i][j]<0)
            {
                cout<<"\n\t\t  Re-enter (instances cannot be negative): ";
                cin>>allocation_matrix[i][j];
            }
        }
    }

    //Input max matrix, NOTE: max_matrix[i][j] >= allocation_matrix[i][j]
    cout<<"\n\t\tEnter the max matrix: ";
    int temp;
    for(int i=0;i<n;++i)
    {
        for(int j=0;j<m;++j)
        {
            cout<<"\n\t\t  Enter the max instances for process "<<process_name[i];
            cout<<" of resource "<<resource_name[j]<<" : ";
            cin>>max_matrix[i][j];
            while(max_matrix[i][j]<0)
            {
```

```
            cout<<"\n\t\t  Re-enter (instances cannot be negative): ";
            cin>>max_matrix[i][j];
        }
        while(max_matrix[i][j]<allocation_matrix[i][j])
        {
            cout<<"\n\t\t  Re-enter (max instances of a resource for a process should be";
            cout<<" greater than or equal to allocated): ";
            cin>>max_matrix[i][j];
        }

    }
}

//Input the available instances of the resources
cout<<"\n\t\tEnter the available instances for each resource: ";
for(int i=0;i<m;++i)
{
    cout<<"\n\t\t  Available intances of the resource "<<resource_name[i]<<" : ";
    cin>>available_array[i];
    while(available_array[i]<0)
    {
        cout<<"\n\t\t  Re-enter (instances cannot be negative): ";
        cin>>available_array[i];
    }
}

//display the table
//Printing tabular representation of the problem
cout<<"\n\t\t\tTabular Representation of problem.";
cout<<"\n\t\t    Allocation      Max         Need  ";
cout<<"\n\n\t\t   ";
for(int i=0;i<3;++i)
{
    for(int j=0;j<m;++j)
    {
        cout<<" "<<resource_name[j]<<" ";
    }
    cout<<" ";
}
cout<<"\n";
for(int i=0;i<n;++i)
{
    cout<<"\n\t\t"<<process_name[i]<<" ";
    for(int j=0;j<m;++j)
    {
        if(allocation_matrix[i][j]/10!=0)
        {
            cout<<allocation_matrix[i][j]<<" ";
        }
        else
        {
            cout<<" "<<allocation_matrix[i][j]<<" ";
        }
    }
    cout<<" ";
    for(int j=0;j<m;++j)
    {
        if(max_matrix[i][j]/10!=0)
        {
            cout<<max_matrix[i][j]<<" ";
        }
        else
        {
            cout<<" "<<max_matrix[i][j]<<" ";
        }
    }
    cout<<" ";
    for(int j=0;j<m;++j)
    {
        if((max_matrix[i][j]-allocation_matrix[i][j])/10!=0)
        {
            cout<<max_matrix[i][j]-allocation_matrix[i][j]<<" ";
        }
        else
        {
            cout<<" "<<max_matrix[i][j]-allocation_matrix[i][j]<<" ";
        }
    }
}

int need_matrix[n][m];
//calculating need matrix
for(int i=0;i<n;++i)
{
    for(int j=0;j<m;++j)
    {
        need_matrix[i][j]=max_matrix[i][j]-allocation_matrix[i][j];
    }
}
```

```
//Initialising work and finish of size m and n
int work[m],finish[n];
for(int i=0;i<m;++i)
{
    work[i]=available_array[i];
}
//make all processes as not complete
for(int i=0;i<n;++i)
{
    finish[i]=0;//not completed processes
}


//Implementation of banker's Algorithm to get to safe state.
int flag=0,inc=0,f=0;
temp=0;
int safe_state[n];
for(int x=0;x<n*n;++x)
{

        for(int j=0;j<m;++j)
        {
            if(need_matrix[temp][j]>work[j])
            {
                flag=1;
                break;
            }
        }
        if(flag==1)
        {
            temp=(temp+1)%n;//incrementing processes again and again.
        }
        else
        {
            if(finish[temp]==0)
            {
                for(int j=0;j<m;++j)
                {
                    work[j]=work[j]+allocation_matrix[temp][j];
                    //Update the work by adding the allocated instances of temp as it is completed/finished
                }
                safe_state[inc]=temp;
                finish[temp]=1;//Process completed/finished
                ++inc;

            }
            temp=(temp+1)%n;
        }

    flag=0;
    if(all_process_completed(n,finish)==1)
    {
        f=1;
        cout<<"\n\n\t\tAll process have successfully executed!!";
        break;
    }

}
if(f==1)
{
    cout<<"\n\t\tDeadlock has been successfully avoided!";
    cout<<"\n\t\t\tThe Safety Sequence is: \n\t\t    ";
        for(int i=0;i<n;++i)
        {
            if(i==n-1)
            {
                cout<<process_name[safe_state[i]];
            }
            else
            {
                cout<<process_name[safe_state[i]]<<" --> ";
            }

        }
    }
    else
    {
        cout<<"\n\t\t\tNo safety Sequence can avoid deadlock!!";
    }


    return 0;
}



/*
Output: (three outputs attached)

Example 1: Deadlock avoidance safety sequence found!
```

OS Lab Assignment
Submitted by: Sanskar Sharma
PRN: 0120180381

Deadlock Avoidance using Banker's Algorithm

```
Enter the number of processes: 5

Enter the name of each process:
    Enter the name of 1 process: P0

    Enter the name of 2 process: P1

    Enter the name of 3 process: P2

    Enter the name of 4 process: P3

    Enter the name of 5 process: P4

Enter the number of resources: 4

Enter the name of each resource:
    Enter the name of 1 resource: A

    Enter the name of 2 resource: B

    Enter the name of 3 resource: C

    Enter the name of 4 resource: D

Enter the allocation matrix:
  Enter the allocated instances for process P0 of resource A : 0

  Enter the allocated instances for process P0 of resource B : 0

  Enter the allocated instances for process P0 of resource C : 1

  Enter the allocated instances for process P0 of resource D : 2

  Enter the allocated instances for process P1 of resource A : 1

  Enter the allocated instances for process P1 of resource B : 0

  Enter the allocated instances for process P1 of resource C : 0

  Enter the allocated instances for process P1 of resource D : 0

  Enter the allocated instances for process P2 of resource A : -3

  Re-enter (instances cannot be negative): 1

  Enter the allocated instances for process P2 of resource B : 3

  Enter the allocated instances for process P2 of resource C : 5

  Enter the allocated instances for process P2 of resource D : 4

  Enter the allocated instances for process P3 of resource A : 0

  Enter the allocated instances for process P3 of resource B : 6

  Enter the allocated instances for process P3 of resource C : 3

  Enter the allocated instances for process P3 of resource D : 2

  Enter the allocated instances for process P4 of resource A : 0

  Enter the allocated instances for process P4 of resource B : 0

  Enter the allocated instances for process P4 of resource C : 1

  Enter the allocated instances for process P4 of resource D : 4

Enter the max matrix:
  Enter the max instances for process P0 of resource A : 0

  Enter the max instances for process P0 of resource B : 0

  Enter the max instances for process P0 of resource C : 1

  Enter the max instances for process P0 of resource D : 2

  Enter the max instances for process P1 of resource A : 1

  Enter the max instances for process P1 of resource B : 7

  Enter the max instances for process P1 of resource C : 5

  Enter the max instances for process P1 of resource D : 0
```

```
Enter the max instances for process P2 of resource A : 2

Enter the max instances for process P2 of resource B : 3

Enter the max instances for process P2 of resource C : 5

Enter the max instances for process P2 of resource D : 6

Enter the max instances for process P3 of resource A : -4

Re-enter (instances cannot be negative): 0

Enter the max instances for process P3 of resource B : 6

Enter the max instances for process P3 of resource C : 5

Enter the max instances for process P3 of resource D : 2

Enter the max instances for process P4 of resource A : 0

Enter the max instances for process P4 of resource B : 6

Enter the max instances for process P4 of resource C : 5

Enter the max instances for process P4 of resource D : 3

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 6

                Enter the available instances for each resource:
                  Available intances of the resource A : 1

                  Available intances of the resource B : 5

                  Available intances of the resource C : 2

                  Available intances of the resource D : 0

                        Tabular Representation of problem.
                     Allocation      Max         Need

                     A  B  C  D   A  B  C  D   A  B  C  D

                P0   0  0  1  2   0  0  1  2   0  0  0  0
                P1   1  0  0  0   1  7  5  0   0  7  5  0
                P2   1  3  5  4   2  3  5  6   1  0  0  2
                P3   0  6  3  2   0  6  5  2   0  0  2  0
                P4   0  0  1  4   0  6  5  6   0  6  4  2

                All process have successfully executed!!
                Deadlock has been successfully avoided!
                       The Safety Sequence is:
                     P0 --> P2 --> P3 --> P4 --> P1
--------------------------------
Process exited after 137.9 seconds with return value 0
Press any key to continue . . .

Example 2: Deadlock avoidance safety sequence found!



                                  OS Lab Assignment
                               Submitted by: Sanskar Sharma
                                    PRN: 0120180381

                       Deadlock Avoidance using Banker's Algorithm
Enter the number of processes: 5

Enter the name of each process:
    Enter the name of 1 process: P0

    Enter the name of 2 process: P1

    Enter the name of 3 process: P2

    Enter the name of 4 process: P3

    Enter the name of 5 process: P4

Enter the number of resources: 3

Enter the name of each resource:
    Enter the name of 1 resource: A

    Enter the name of 2 resource: B

    Enter the name of 3 resource: C

Enter the allocation matrix:
  Enter the allocated instances for process P0 of resource A : 0
```

```
Enter the allocated instances for process P0 of resource B : 1

Enter the allocated instances for process P0 of resource C : 0

Enter the allocated instances for process P1 of resource A : 2

Enter the allocated instances for process P1 of resource B : 0

Enter the allocated instances for process P1 of resource C : 0

Enter the allocated instances for process P2 of resource A : 3

Enter the allocated instances for process P2 of resource B : 0

Enter the allocated instances for process P2 of resource C : 2

Enter the allocated instances for process P3 of resource A : 2

Enter the allocated instances for process P3 of resource B : 1

Enter the allocated instances for process P3 of resource C : 1

Enter the allocated instances for process P4 of resource A : 0

Enter the allocated instances for process P4 of resource B : 0

Enter the allocated instances for process P4 of resource C : 2

Enter the max matrix:
  Enter the max instances for process P0 of resource A : 7

  Enter the max instances for process P0 of resource B : 5

  Enter the max instances for process P0 of resource C : 3

  Enter the max instances for process P1 of resource A : 3

  Enter the max instances for process P1 of resource B : 2

  Enter the max instances for process P1 of resource C : 2

  Enter the max instances for process P2 of resource A : 9

  Enter the max instances for process P2 of resource B : 0

  Enter the max instances for process P2 of resource C : 2

  Enter the max instances for process P3 of resource A : 2

  Enter the max instances for process P3 of resource B : 2

  Enter the max instances for process P3 of resource C : 2

  Enter the max instances for process P4 of resource A : 4

  Enter the max instances for process P4 of resource B : 3

  Enter the max instances for process P4 of resource C : 3

Enter the available instances for each resource:
  Available intances of the resource A : 3

  Available intances of the resource B : 3

  Available intances of the resource C : 2

        Tabular Representation of problem.
    Allocation      Max         Need

    A  B  C   A  B  C   A  B  C

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

All process have successfully executed!!
Deadlock has been successfully avoided!
        The Safety Sequence is:
    P1 --> P3 --> P4 --> P0 --> P2
--------------------------------
Process exited after 61.87 seconds with return value 0
Press any key to continue . . .

Example 3: Unsafe states/ no safety sequence.
```

Submitted by: Sanskar Sharma
PRN: 0120180381

```
        Deadlock Avoidance using Banker's Algorithm
Enter the number of processes: 5

Enter the name of each process:
    Enter the name of 1 process: P1

    Enter the name of 2 process: P2

    Enter the name of 3 process: P3

    Enter the name of 4 process: P4

    Enter the name of 5 process: P5

Enter the number of resources: 3

Enter the name of each resource:
    Enter the name of 1 resource: A

    Enter the name of 2 resource: B

    Enter the name of 3 resource: C

Enter the allocation matrix:
  Enter the allocated instances for process P1 of resource A : 0

  Enter the allocated instances for process P1 of resource B : 1

  Enter the allocated instances for process P1 of resource C : 0

  Enter the allocated instances for process P2 of resource A : 2

  Enter the allocated instances for process P2 of resource B : 0

  Enter the allocated instances for process P2 of resource C : 0

  Enter the allocated instances for process P3 of resource A : 3

  Enter the allocated instances for process P3 of resource B : 0

  Enter the allocated instances for process P3 of resource C : 3

  Enter the allocated instances for process P4 of resource A : 2

  Enter the allocated instances for process P4 of resource B : 1

  Enter the allocated instances for process P4 of resource C : 1

  Enter the allocated instances for process P5 of resource A : 0

  Enter the allocated instances for process P5 of resource B : 0

  Enter the allocated instances for process P5 of resource C : 2

Enter the max matrix:
  Enter the max instances for process P1 of resource A : 0

  Enter the max instances for process P1 of resource B : 0
Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 0

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 5

Enter the max instances for process P1 of resource C : 3

Enter the max instances for process P2 of resource A : 3

Enter the max instances for process P2 of resource B : 2

Enter the max instances for process P2 of resource C : 2

Enter the max instances for process P3 of resource A : 9

Enter the max instances for process P3 of resource B : 0

Enter the max instances for process P3 of resource C : 2

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 2

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 2

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 2

Re-enter (max instances of a resource for a process should be greater than or equal to allocated): 5

Enter the max instances for process P4 of resource A : 5
```

```
  Enter the max instances for process P4 of resource B : 5

  Enter the max instances for process P4 of resource C : 4

  Enter the max instances for process P5 of resource A : 3

  Enter the max instances for process P5 of resource B : 3

  Enter the max instances for process P5 of resource C : 9

Enter the available instances for each resource:
  Available intances of the resource A : 1

  Available intances of the resource B : 1

  Available intances of the resource C : 1

        Tabular Representation of problem.
    Allocation      Max         Need

    A  B  C   A  B  C   A  B  C

P1  0  1  0   0  5  3   0  4  3
P2  2  0  0   3  2  2   1  2  2
P3  3  0  3   9  0  5   6  0  2
P4  2  1  1   5  5  4   3  4  3
P5  0  0  2   3  3  9   3  3  7
       No safety Sequence can avoid deadlock!!
--------------------------------
Process exited after 96.96 seconds with return value 0
Press any key to continue . . .
*/
```