**Assignment 4**

**Name: Yadnesh Patil**

**Div: CS-C**

**Roll No: 23**

**Problems Statement:** Write a program in C or C++ or Java for Bankers algorithm for avoidance of deadlock.

**Code:**

#include <bits/stdc++.h>

using namespace std;

void Calculate();

void IMP();

int allocation[10][3], need[10][3], Max[10][3], available[10][3];

int p, current[3];

bool executed[10], come;

int main()

{

    int keepon = 1;

    cout << "Enter No. of processes: ";

    cin >> p;

    cout << "\n";

    cout << "Enter the current resources: ";

    cin >> current[0] >> current[1] >> current[2];

    for (int i = 0; i < p; ++i)

    {

        cout << "\n\n\t\t\tProcess P" << i + 1 << " Details\n";

        cout << "Enter Allocation :";

        cin >> allocation[i][0] >> allocation[i][1] >> allocation[i][2];

        cout << "Enter Max :";

        cin >> Max[i][0] >> Max[i][1] >> Max[i][2];

        need[i][0] = Max[i][0] - allocation[i][0];

        need[i][1] = Max[i][1] - allocation[i][1];

        need[i][2] = Max[i][2] - allocation[i][2];

    }

    cout << "\n\n\t\t\tTable for Bankers Algo\n\n";

    cout << "Initial Resources: " << current[0] << "" << current[1] << "" << current[2] << "\n\n";

    cout << "Process    Max    Allocation    Need\n";

    for (int i = 0; i < p; ++i)

    {

        cout << "  P" << i + 1 << "    ";

        cout << "  " << Max[i][0] << "" << Max[i][1] << "" << Max[i][2] << "     ";

        cout << "" << allocation[i][0] << "" << allocation[i][1] << "" << allocation[i][2] << "     ";

        cout << "" << need[i][0] << "" << need[i][1] << "" << need[i][2];

        cout << "\n";

    }

    cout << "\n\n";

    Calculate();

    while (keepon)

    {

        int val, pro;

        cout << "\n\nSelect Below oprations:\n\n";

        cout << "1.Change Max of process: \n";

        cout << "2.Change Allocation of process\n";

        cout << "3.Change Initial Resources\n";

        cout << "4.Exit\n\n";

        cin >> val;

        if (val == 1)

        {

            cout << "\n\nEnter Process No: ";

            cin >> pro;

            cout << "\nEnter New Max: ";

            cin >> Max[pro - 1][0] >> Max[pro - 1][1] >> Max[pro - 1][2];

        }

        else if(val == 2)

        {

            cout << "\n\nEnter Process No: ";

            cin >> pro;

            cout << "\nEnter New Allocation: ";

            cin >> allocation[pro - 1][0] >> allocation[pro - 1][1] >> allocation[pro - 1][2];

        }

        else if(val == 3)

        {

            cout << "\nEnter Initial Resources: ";

            cin >> current[0] >> current[1] >> current[2];

        }

        else

        {

            break;

        }

        Calculate();

    }

    return 0;

}

void Calculate()

{

    IMP();

    int i, j;

    for (i = 0; i < p; ++i)

    {

        for (j = 0; j < p; ++j)

        {

            while (executed[j] && j < p - 1)

            {

                j++;

            }

            if (need[j][0] <= current[0] && need[j][1] <= current[1] && need[j][2] <= current[2])

            {

                if (!executed[j])

                {

                    executed[j] = true;

                    current[0] += allocation[j][0];

                    current[1] += allocation[j][1];

                    current[2] += allocation[j][2];

                    cout << "\nProcess P" << j + 1;

                    cout << "\nCurrent: " << current[0] << "" << current[1] << "" << current[2] << "\n";

                    cout << "\nProcess executed without deadlock";

                    come = true;

                    break;

                }

            }

        }

        if (!come)

        {

            cout << "\n\t\t\tDead lock\n\n";

            break;

        }

        else

        {

            come = false;

        }

    }

}

void IMP()

{

    come = false;

    for (int i = 0; i < 10; ++i)

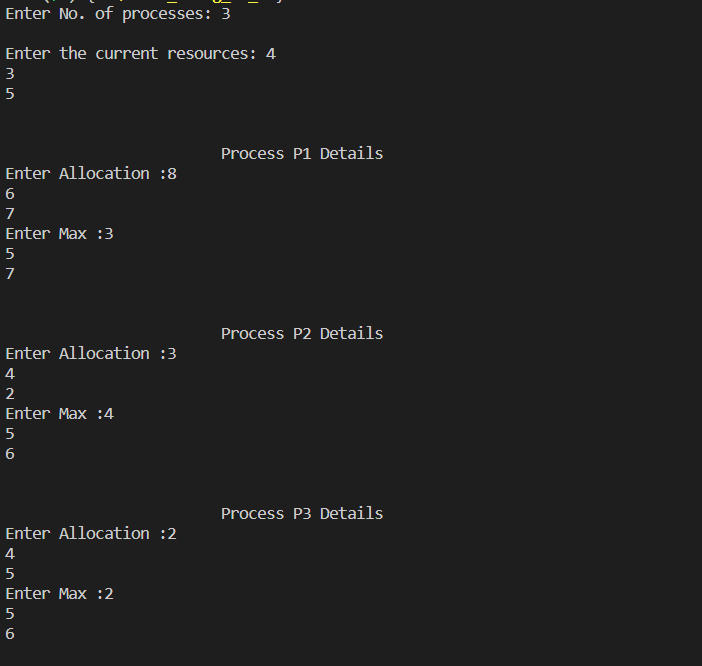
    {

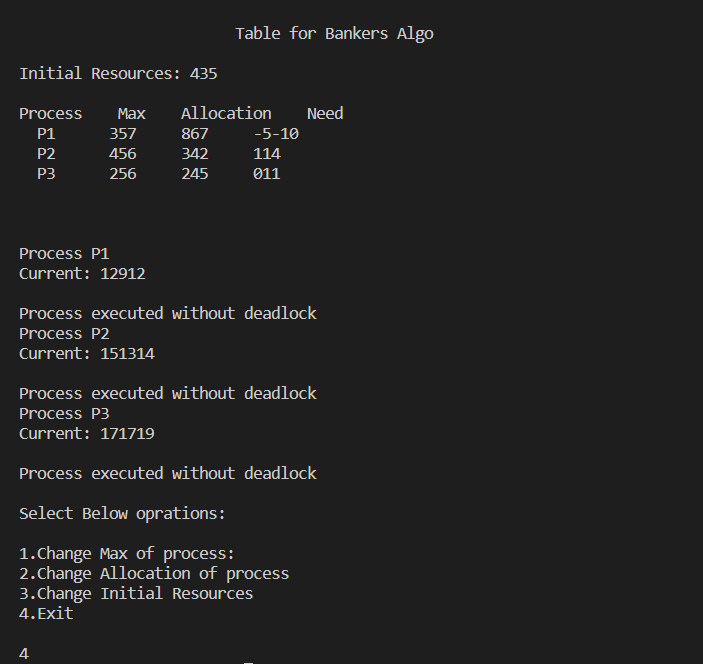
        executed[i] = false;

    }

}

**Output:**

****

****