Experiment Number: 8

Problem Statement: **Write a program to check whether given system is in safe state or not using Banker’s Deadlock Avoidance algorithm.**

NAME: Aadesh Chawla ROLLNO: 12

CLASS: TY-IT-A BATCH: B1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:**

#include <iostream>

using namespace std;

int main()

{

    int numProcesses, numResources, i, j, k;

    numProcesses = 5;

    numResources = 3;

    int allocation[5][3] = { { 0, 1, 0 },

                            { 2, 0, 0 },

                            { 3, 0, 2 },

                            { 2, 1, 1 },

                            { 0, 0, 2 } };

    int maxDemand[5][3] = { { 7, 5, 3 },

                            { 3, 2, 2 },

                            { 9, 0, 2 },

                            { 2, 2, 2 },

                            { 4, 3, 3 } };

    int availableResources[3] = { 3, 3, 2 };

    int finished[5], safeSequence[5], index = 0;

    for (k = 0; k < numProcesses; k++) {

        finished[k] = 0;

    }

    int resourceNeed[5][3];

    for (i = 0; i < numProcesses; i++) {

        for (j = 0; j < numResources; j++)

            resourceNeed[i][j] = maxDemand[i][j] - allocation[i][j];

    }

    int y = 0;

    for (k = 0; k < 5; k++) {

        for (i = 0; i < numProcesses; i++) {

            if (finished[i] == 0) {

                int flag = 0;

                for (j = 0; j < numResources; j++) {

                    if (resourceNeed[i][j] > availableResources[j]){

                        flag = 1;

                        break;

                    }

                }

                if (flag == 0) {

                    safeSequence[index++] = i;

                    for (y = 0; y < numResources; y++)

                        availableResources[y] += allocation[i][y];

                    finished[i] = 1;

                }

            }

        }

    }

    int flag = 1;

    // To check if the sequence is safe or not

    for(int i = 0; i < numProcesses; i++) {

        if(finished[i] == 0) {

            flag = 0;

            cout << "The given sequence is not safe";

            break;

        }

    }

    if(flag == 1) {

        cout << "Following is the SAFE Sequence" << endl;

        for (i = 0; i < numProcesses - 1; i++)

            cout << " P" << safeSequence[i] << " ->";

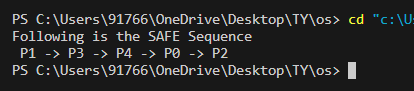
        cout << " P" << safeSequence[numProcesses - 1] << endl;

    }

    return (0);

}

**Output:**

****