**LAB 8 - BANKERS ALGORITHM**

**ADIT LUHADIA**

**190911112**

**IT A**

1. Develop a program to simulate banker’s algorithm. (Consider safety and resourcerequest algorithms)

// 190911112

// Adit Luhadia

#include <iostream>

using namespace std;

int searchP(int \*\*allocation, int \*\*NEED, int available[], bool finished[], int pnum, int rnum)

{

    int flag;

    for (int i = 0; i < pnum; i++)

    {

        if (finished[i])

            continue;

        flag = 0;

        for (int j = 0; j < rnum; j++)

        {

            if (NEED[i][j] > available[j])

            {

                flag = 1;

                break;

            }

        }

        if (flag == 0)

        {

            for (int k = 0; k < rnum; k++)

                available[k] += allocation[i][k];

            finished[i] = true;

            return i;

        }

    }

    return -1;

}

bool safetyCheck(int \*\*allocation, int \*\*NEED, int available[], bool finished[], int pnum, int rnum)

{

    int temp = pnum, safestate[pnum], k = 0, current;

    while (temp-- != 0)

    {

        current = searchP(allocation, NEED, available, finished, pnum, rnum);

        if (current == -1)

        {

            cout << "\nSystem in unsafe state" << endl;

            return false;

        }

        else

        {

            safestate[k++] = current;

        }

    }

    cout << "\nThe safe sequence is" << endl;

    for (int i = 0; i < pnum; i++)

        cout << "p" << safestate[i] << " ";

    return true;

}

bool resourceCheck(int \*\*allocation, int \*\*NEED, int available[], bool finished[], int pnum, int rnum)

{

    int pid, request[rnum], flag;

    cout << "\nEnter the process id" << endl;

    cin >> pid;

    cout << "\nEnter the request for each resource" << endl;

    for (int i = 0; i < rnum; i++)

        cin >> request[i];

    for (int j = 0; j < rnum; j++)

    {

        if (request[j] > NEED[pid][j])

            return false;

    }

    for (int j = 0; j < rnum; j++)

    {

        if (request[j] > available[j])

            return false;

    }

    for (int j = 0; j < rnum; j++)

    {

        NEED[pid][j] -= request[j];

        available[j] -= request[j];

        allocation[pid][j] += request[j];

    }

    if (safetyCheck(allocation, NEED, available, finished, pnum, rnum))

        return true;

    else

        return false;

}

int main()

{

    cout << "190911112\nAdit Luhadia";

    int pnum, rnum;

    cout << "\nEnter the number of processes and the number of resource types" << endl;

    cin >> pnum >> rnum;

    int \*allocation[pnum], \*MAX[pnum], \*NEED[pnum], available[rnum], safestate[pnum], k = 0, temp = pnum, current;

    bool finished[pnum];

    for (int i = 0; i < pnum; i++)

        finished[i] = false;

    for (int i = 0; i < pnum; i++)

    {

        allocation[i] = new int[rnum];

        MAX[i] = new int[rnum];

        NEED[i] = new int[rnum];

    }

    cout << "\nEnter the allocation matrix" << endl;

    for (int i = 0; i < pnum; i++)

    {

        for (int j = 0; j < rnum; j++)

            cin >> allocation[i][j];

    }

    cout << "\nEnter the MAX matrix" << endl;

    for (int i = 0; i < pnum; i++)

    {

        for (int j = 0; j < rnum; j++)

        {

            cin >> MAX[i][j];

            NEED[i][j] = MAX[i][j] - allocation[i][j];

        }

    }

    cout << "\nEnter the number of resources available for each type" << endl;

    for (int i = 0; i < rnum; i++)

        cin >> available[i];

    char choice, cont = 'Y';

    while (cont == 'Y')

    {

        cout << "\nEnter choice:\n'S':Safety Algorithm\n'R':Request Allocation Algorithm" << endl;

        cin >> choice;

        switch (choice)

        {

        case 'S':

            safetyCheck(allocation, NEED, available, finished, pnum, rnum);

            break;

        case 'R':

            if (resourceCheck(allocation, NEED, available, finished, pnum, rnum))

                cout << "\nRequest can be immediately granted" << endl;

            else

                cout << "\nRequest cannot be granted" << endl;

            break;

        default:

            cout << "\nInvalid choice";

        }

        cout << "\nEnter 'Y' to continue and 'N' to exit";

        cin >> cont;

    }

}

Text

Description automatically generated