**Name: Srinivas Sanjeevkumar Chenna**

**Division: A**

**Roll Number: 48**

**Subject: Operating System (OS) LAB**

**Lab Assignment 5**

**Implement following algorithms**

**1. Deadlock Avoidance**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int arrmax[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n, r;

void input()

{

int i, j;

cout << "Enter the no of Processes\t";

cin >> n;

cout << "Enter the no of resource instances\t";

cin >> r;

cout << "Enter the Max Matrix\n";

for (i = 0; i < n; i++)

{

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

{

cin >> arrmax[i][j];

}

}

cout << "Enter the Allocation Matrix\n";

for (i = 0; i < n; i++)

{

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

{

cin >> alloc[i][j];

}

}

cout << "Enter the available Resources\n";

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

{

cin >> avail[j];

}

}

void show()

{

int i, j;

cout << "Process\t Allocation\t Max\t Available\t";

for (i = 0; i < n; i++)

{

cout << "\nP" << i + 1 << "\t ";

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

{

cout << alloc[i][j] << " ";

}

cout << "\t\t";

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

{

cout << arrmax[i][j] << " ";

}

cout << "\t ";

if (i == 0)

{

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

cout << avail[j] << " ";

}

}

}

void cal()

{

int finish[100], temp, need[100][100], flag = 1, k, c1 = 0;

int dead[100];

int safe[100];

int i, j;

for (i = 0; i < n; i++)

{

finish[i] = 0;

}

for (i = 0; i < n; i++)

{

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

{

need[i][j] = arrmax[i][j] - alloc[i][j];

}

}

while (flag)

{

flag = 0;

for (i = 0; i < n; i++)

{

int c = 0;

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

{

if ((finish[i] == 0) && (need[i][j] <= avail[j]))

{

c++;

if (c == r)

{

for (k = 0; k < r; k++)

{

avail[k] += alloc[i][j];

finish[i] = 1;

flag = 1;

}

//cout<<"\nP%d",i;

if (finish[i] == 1)

{

i = n;

}

}

}

}

}

}

j = 0;

flag = 0;

for (i = 0; i < n; i++)

{

if (finish[i] == 0)

{

dead[j] = i;

j++;

flag = 1;

}

}

if (flag == 1)

{

cout << "\n\nDeadlock is present\n";

}

else

{

cout << "\nNo Deadlock present";

}

}

int main()

{

int i, j;

input();

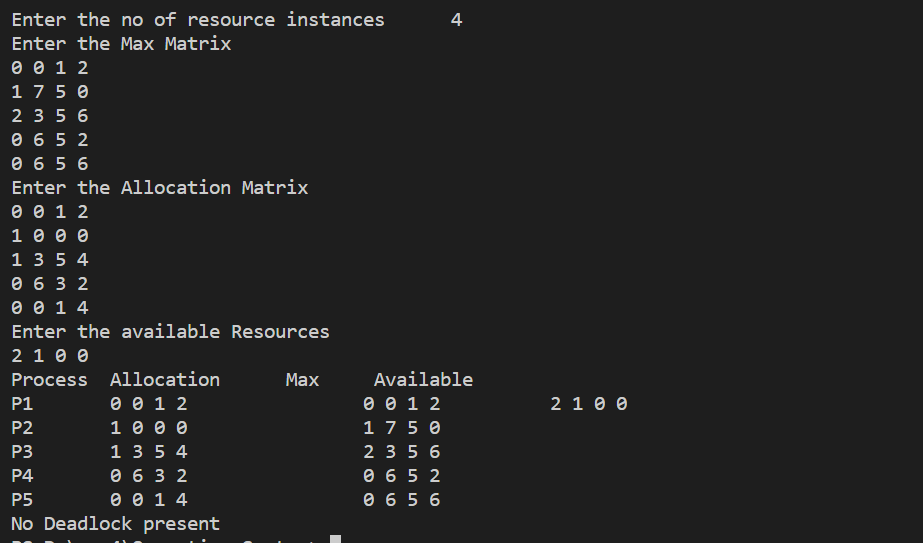
show();

cal();

return 0;

}

**Output:**



**2. Deadlock Detection**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int arrmax[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n, r;

void input()

{

int i, j;

cout << "Enter the no of Processes\t";

cin >> n;

cout << "Enter the no of resource instances\t";

cin >> r;

cout << "Enter the Need Matrix\n";

for (i = 0; i < n; i++)

{

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

{

cin >> need[i][j];

}

}

cout << "Enter the Allocation Matrix\n";

for (i = 0; i < n; i++)

{

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

{

cin >> alloc[i][j];

}

}

cout << "Enter the available Resources\n";

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

{

cin >> avail[j];

}

}

void show()

{

int i, j;

cout << "Process\t Allocation\t Max\t Available\t";

for (i = 0; i < n; i++)

{

cout << "\nP" << i + 1 << "\t ";

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

{

cout << alloc[i][j] << " ";

}

cout << "\t\t";

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

{

cout << arrmax[i][j] << " ";

}

cout << "\t ";

if (i == 0)

{

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

cout << avail[j] << " ";

}

}

}

void cal()

{

int finish[100], temp, need[100][100], flag = 1, k, c1 = 0;

int dead[100];

int safe[100];

int i, j;

for (i = 0; i < n; i++)

{

finish[i] = 0;

}

while (flag)

{

flag = 0;

for (i = 0; i < n; i++)

{

int c = 0;

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

{

if ((finish[i] == 0) && (need[i][j] <= avail[j]))

{

c++;

if (c == r)

{

for (k = 0; k < r; k++)

{

avail[k] += alloc[i][j];

finish[i] = 1;

flag = 1;

}

//cout<<"\nP%d",i;

if (finish[i] == 1)

{

i = n;

}

}

}

}

}

}

j = 0;

flag = 0;

for (i = 0; i < n; i++)

{

if (finish[i] == 0)

{

dead[j] = i;

j++;

flag = 1;

}

}

if (flag == 1)

{

cout << "\n\nSystem is in Deadlock and the Deadlock process are\n";

for (i = 0; i < n; i++)

{

cout << "P" << dead[i] << "\t";

}

}

else

{

cout << "\nNo Deadlock Occur";

}

}

int main()

{

int i, j;

cout << "\*\*\*\*\*\*\*\*\*\* Deadlock Detection Algorithm \*\*\*\*\*\*\*\*\*\*\*\*\n";

input();

show();

cal();

return 0;

}

**Output:**

