ASSIGNMENT 5

PROGRAM

#include <iostream>

using namespace std;

const int MAX\_P = 10; // Max number of processes

const int MAX\_R = 10; // Max number of resource types

int main() {

int n, m; // n = number of processes, m = number of resources

int allocation[MAX\_P][MAX\_R];

int maxNeed[MAX\_P][MAX\_R];

int need[MAX\_P][MAX\_R];

int available[MAX\_R];

int work[MAX\_R];

bool finished[MAX\_P] = {false};

int safeSequence[MAX\_P];

int count = 0;

// Input

cout << "Enter number of processes: ";

cin >> n;

cout << "Enter number of resources: ";

cin >> m;

cout << "\nEnter Allocation Matrix:\n";

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

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

cin >> allocation[i][j];

cout << "\nEnter Maximum Need Matrix:\n";

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

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

cin >> maxNeed[i][j];

cout << "\nEnter Available Resources:\n";

for (int i = 0; i < m; i++)

cin >> available[i];

// Calculate Need Matrix = Max - Allocation

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

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

need[i][j] = maxNeed[i][j] - allocation[i][j];

// Initialize work = available

for (int i = 0; i < m; i++)

work[i] = available[i];

// Banker's Algorithm

while (count < n) {

bool found = false;

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

if (!finished[i]) {

bool canAllocate = true;

for (int j = 0; j < m; j++) {

if (need[i][j] > work[j]) {

canAllocate = false;

break;

}

}

if (canAllocate) {

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

work[j] += allocation[i][j];

safeSequence[count++] = i;

finished[i] = true;

found = true;

}

}

}

if (!found) {

cout << "\nThe system is NOT in a safe state." << endl;

return 0;

}

}

// Safe state found

cout << "\nThe system is in a SAFE state.\nSafe sequence: ";

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

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

cout << endl;

return 0;

}

OUTPUT:

PS C:\Users\User\Desktop\OSL> g++ osl5.cpp

PS C:\Users\User\Desktop\OSL> .\a

Enter number of processes: 5

Enter number of resources: 3

Enter Allocation Matrix:

0

1

0

2

0

0

3

0

2

2

1

1

0

0

2

Enter Maximum Need Matrix:

7

5

3

3

2

2

9

0

2

2

2

2

4

3

3

Enter Available Resources:

3

3

2

The system is in a SAFE state.

Safe sequence: P1 P3 P4 P0 P2

PS C:\Users\User\Desktop\OSL> .\a

Enter number of processes: 3

Enter number of resources: 2

Enter Allocation Matrix:

0

1

2

0

3

0

Enter Maximum Need Matrix:

7

5

3

2

9

0

Enter Available Resources:

2

0

The system is NOT in a safe state.

PS C:\Users\User\Desktop\OSL>