TASK :

#include<iostream>

#include<cstdlib>

#include<ctime>

using namespace std;

const int no\_of\_processes = 5;

const int no\_of\_resources = 3;

void print\_state(bool finish[], int work[], int allocation[][no\_of\_resources], int need[][no\_of\_resources])

{

cout << "Process\t\tMaximum\t\tAllocated\tNeed\t\tFinish\n";

cout << "\t\tR1 R2 R3\tR1 R2 R3\tR1 R2 R3\n";

for (int i = 0; i < no\_of\_processes; i++)

{

cout << "P" << i << "\t\t";

for (int j = 0; j < no\_of\_resources; j++)

{

cout << need[i][j] + allocation[i][j] << " ";

}

cout << "\t\t";

for (int j = 0; j < no\_of\_resources; j++)

{

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

}

cout << "\t\t";

for (int j = 0; j < no\_of\_resources; j++)

{

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

}

cout << "\t\t" << (finish[i] ? "Finish" : "Not finish") << endl;

}

cout << "Available resources: " << work[0] << " " << work[1] << " " << work[2] << endl;

}

bool is\_safe\_state(bool finish[], int work[], int allocation[][no\_of\_resources], int need[][no\_of\_resources], int &next\_process)

{

bool found = false;

for (int i = next\_process; i < no\_of\_processes; i = (i + 1) % no\_of\_processes)

{

bool possible = true;

for (int j = 0; j < no\_of\_resources; j++)

{

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

{

possible = false;

break;

}

}

if (possible && !finish[i])

{

found = true;

next\_process = (i + 1) % no\_of\_processes;

for (int j = 0; j < no\_of\_resources; j++)

{

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

}

finish[i] = true;

break;

}

}

return found;

}

int main()

{

int max\_demand[no\_of\_processes][no\_of\_resources] = {

{7, 5, 3},

{3, 2, 2},

{9, 0, 2},

{2, 2, 2},

{4, 3, 3}

};

int allocation[no\_of\_processes][no\_of\_resources] = {

{0, 1, 0},

{2, 0, 0},

{3, 0, 2},

{2, 1, 1},

{0, 0, 2}

};

int work[no\_of\_resources] = { 3, 3, 2 };

int need[no\_of\_processes][no\_of\_resources];

bool finish[no\_of\_processes] = { false };

int next\_process = 0;

// Calculate the need matrix

for (int i = 0; i < no\_of\_processes; i++)

{

for (int j = 0; j < no\_of\_resources; j++)

{

need[i][j] = max\_demand[i][j] - allocation[i][j];

}

}

// Run the Banker's Algorithm

cout << "Initial state:\n";

print\_state(finish, work, allocation, need);

cout << endl;

cout << "Sequence of processes:\n";

int count = 0;

while (count < no\_of\_processes)

{

int process\_id;

bool safe = is\_safe\_state(finish, work, allocation, need, next\_process);

if (safe)

{

process\_id = (next\_process + no\_of\_processes - 1) % no\_of\_processes;

cout << "P" << process\_id << " -> ";

count++;

}

else

{

cout << "Unsafe state: no process can proceed\n";

break;

}

}

cout << "End\n";

return 0;

}

