Name : Ruturaj Sandip Sutar Roll No : 59 Div : B

Batch : 2 PRN:-12310720

Implementation of deadlock detection

1. **Safety Algorithm**

Code:-

```
#include <iostream>
#include <vector>
using namespace std;
bool isSafe(vector<vector<int>> &allocation, vector<vector<int>> &need, vector<int> &available) {
   int numProcesses = allocation.size();
   int numResources = available.size();

   vector<bool> finish(numProcesses, false);
   vector<int> safeSequence(numProcesses);
   vector<int> work = available;

   int count = 0;
   while (count < numProcesses) {
      bool found = false;
      for (int i = 0; i < numProcesses; i++) {
         if (!finish[i]) {
            bool possible = true;
            for (int j = 0; j < numResources; j++) {
               if (need[i][j] > work[j]) {
                  possible = false;
                  break;
               }
```

```
            }
            if (possible) {
                for (int j = 0; j < numResources; j++) {
                    work[j] += allocation[i][j];
                }
                safeSequence[count++] = i;
                finish[i] = true;
                found = true;
            }
        }
    }
    if (!found) {
        cout << "System is not in a safe state.\n";
        return false;
    }
  }
  cout << "System is in a safe state.\nSafe Sequence: ";
  for (int i = 0; i < numProcesses; i++) {
    cout << safeSequence[i] << " ";
  }
  cout << endl;

  return true;
}

int main() {
  int numProcesses, numResources;
  cout << "Enter the number of processes: ";
  cin >> numProcesses;

  cout << "Enter the number of resources: ";
```

```
cin >> numResources;
vector<vector<int>> allocation(numProcesses, vector<int>(numResources));
vector<vector<int>> maximum(numProcesses, vector<int>(numResources));
vector<vector<int>> need(numProcesses, vector<int>(numResources));
vector<int> available(numResources);

cout << "Enter the allocation matrix (process-wise):\n";
for (int i = 0; i < numProcesses; i++) {
   for (int j = 0; j < numResources; j++) {
      cin >> allocation[i][j];
   }
}
cout << "Enter the maximum matrix (process-wise):\n";
for (int i = 0; i < numProcesses; i++) {
   for (int j = 0; j < numResources; j++) {
      cin >> maximum[i][j];
   }
}

cout << "Enter the available resources:\n";
for (int i = 0; i < numResources; i++) {
   cin >> available[i];
}
for (int i = 0; i < numProcesses; i++) {
   for (int j = 0; j < numResources; j++) {
      need[i][j] = maximum[i][j] - allocation[i][j];
   }
}
isSafe(allocation, need, available);

return 0;
```

}

Output:-

Enter the number of processes: 5

Enter the number of resources: 3

Enter the allocation matrix (process-wise):

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Enter the maximum matrix (process-wise):

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter the available resources:

3 3 2

System is in a safe state.

Safe Sequence: 1 3 4 0 2