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

**Batch : 2 PRN:-12310720**

**Banker’s Algorithm**

1. **Resource Request Algorithm**

**Code:-**

*#include <iostream>*

*#include <vector>*

*using namespace std;*

*void calculateNeed(int need[][10], int max[][10], int allocation[][10], int processes, int resources) {*

*for (int i = 0; i < processes; i++) {*

*for (int j = 0; j < resources; j++) {*

*need[i][j] = max[i][j] - allocation[i][j];*

*}*

*}*

*}*

*bool isSafe(int processes, int resources, int allocation[][10], int max[][10], int available[]) {*

*int need[10][10];*

*calculateNeed(need, max, allocation, processes, resources);*

*bool finished[10] = {false};*

*int safeSequence[10];*

*int work[10];*

*for (int i = 0; i < resources; i++)*

*work[i] = available[i];*

*int count = 0;*

*while (count < processes) {*

*bool found = false;*

*for (int p = 0; p < processes; p++) {*

*if (!finished[p]) {*

*int j;*

*for (j = 0; j < resources; j++)*

*if (need[p][j] > work[j])*

*break;*

*if (j == resources) {*

*for (int k = 0; k < resources; k++)*

*work[k] += allocation[p][k];*

*safeSequence[count++] = p;*

*finished[p] = true;*

*found = true;*

*}*

*}*

*}*

*if (!found) {*

*return false;*

*}*

*}*

*return true;*

*}*

*bool requestResources(int process, int processes, int resources, int allocation[][10], int max[][10], int available[], int request[]) {*

*int need[10][10];*

*calculateNeed(need, max, allocation, processes, resources);*

*for (int i = 0; i < resources; i++) {*

*if (request[i] > need[process][i]) {*

*cout << "Error: Process has exceeded its maximum claim." << endl;*

*return false;*

*}*

*}*

*for (int i = 0; i < resources; i++) {*

*if (request[i] > available[i]) {*

*cout << "Process " << process << " must wait. Not enough resources available." << endl;*

*return false;*

*}*

*}*

*for (int i = 0; i < resources; i++) {*

*available[i] -= request[i];*

*allocation[process][i] += request[i];*

*need[process][i] -= request[i];*

*}*

*if (isSafe(processes, resources, allocation, max, available)) {*

*cout << "Request can be granted. System remains in a safe state." << endl;*

*return true;*

*} else {*

*for (int i = 0; i < resources; i++) {*

*available[i] += request[i];*

*allocation[process][i] -= request[i];*

*need[process][i] += request[i];*

*}*

*cout << "Request cannot be granted. System would be unsafe." << endl;*

*return false;*

*}*

*}*

*int main() {*

*int processes, resources;*

*cout << "Enter number of processes: ";*

*cin >> processes;*

*cout << "Enter number of resources: ";*

*cin >> resources;*

*int allocation[10][10], max[10][10], available[10];*

*cout << "Enter Allocation Matrix:\n";*

*for (int i = 0; i < processes; i++)*

*for (int j = 0; j < resources; j++)*

*cin >> allocation[i][j];*

*cout << "Enter Max Matrix:\n";*

*for (int i = 0; i < processes; i++)*

*for (int j = 0; j < resources; j++)*

*cin >> max[i][j];*

*cout << "Enter Available Resources:\n";*

*for (int i = 0; i < resources; i++)*

*cin >> available[i];*

*if (!isSafe(processes, resources, allocation, max, available)) {*

*cout << "Initial state is unsafe!" << endl;*

*return 0;*

*}*

*int process;*

*cout << "Enter the process number making the request: ";*

*cin >> process;*

*int request[10];*

*cout << "Enter the resource request by process " << process << ":\n";*

*for (int i = 0; i < resources; i++)*

*cin >> request[i];*

*requestResources(process, processes, resources, allocation, max, available, request);*

*return 0;*

*}*

**Output:-**

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 Max Matrix:

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter Available Resources:

3 3 2

Enter the process number making the request: 1

Enter the resource request by process 1:

1 0 2

Request can be granted. System remains in a safe state.