## Design, develop and implement a C/C++/Java program to implement Banker’s algorithm. Assume suitable input required to demonstrate the results..

**PROGRAM**

import java.util.Scanner;

public class Bankers {

int max[][], need[][], available [][], allocation [][];

int np, nr;

boolean flag;

Scanner sc = new Scanner(System.in);

public void input() {

System.out.println("Enter the number of process: ");

np = sc.nextInt();

System.out.println("Enter the number of resources: ");

nr = sc.nextInt();

max = new int[np][nr];

need = new int[np][nr];

available = new int[1][nr];

allocation = new int[np][nr];

System.out.println("Enter the allocation matrix: ");

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

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

allocation[i][j] = sc.nextInt();

}

}

System.out.println("Enter the max matrix: ");

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

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

max[i][j] = sc.nextInt();

}

}

System.out.println("Enter the available matrix: ");

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

available[0][i] = sc.nextInt();

}

sc.close();

}

public void cal\_need() {

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

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

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

}

}

}

public boolean check(int p) {

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

if(need[p][i]<= available[0][i] ) {

flag=true;

}

else {

flag = false;

break;

}

}

return flag;

}

public void algorithm() {

cal\_need();

int c = 0;

boolean status[] = new boolean[np];

while(c<np) {

boolean allocated = false;

boolean ret;

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

ret = check(i);

if(!status[i] && ret) {

status[i] = true;

allocated =true;

c++;

System.out.println("Allocated Process: " +i);

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

available[0][j] = available[0][j] + allocation[i][j];

}

}

}

if(!allocated) break;

}

if(c == np)

System.out.println("\n Safely Allocated.");

else

System.out.println("Unsafe allocation");

}

public static void main(String args[]) {

Bankers obj = new Bankers();

obj.input();

obj.algorithm();

}

}