VIT-AP UNIVERSITY

School: SCOPE
Semester: WIN 2020-21
Subject: Operating System Lab
Subject Code: CSE2008

Assignment 7

NAME:MAJJIGA JASWANTH

Registration No:20BCD7171

QUESTION:

1. Implement the deadlock prevention using banker's algorithm. Calculate and display the Allocated matrix, Max, need matrix, the allotted process and the Available matrix after completion of every process, from the following inputs:

Enter number of process: 5

Enter number of resources: 3

Enter total numbers of each resources: 10 5 7

Enter Max resources for each process:

for process 1: 7 5 3

for process 2: 3 2 2

for process 3: 9 0 2

for process 4: 2 2 2

for process 5: 4 3 3

Enter allocated resources for each process:

for process 1: 0 1 0

for process 2: 3 0 2

for process 3: 3 0 2

for process 4: 2 1 1

for process 5: 0 0 2

Available resources: 2 3 0

Finally, identify whether the system is in a safe state or not.

CODE:

```
import java.util.*;
import java.io.*;
import java.util.Scanner;
class Main {
static void findNeedValue(int needArray[][],
int maxArray[][],
int allocationArray[][],
int totalProcess,
int totalResources)
{
        for (int i = 0 ; i < totalProcess ; i++)
        {
        for (int j = 0 ; j < totalResources ; j++){
        needArray[i][j] = maxArray[i][j] - allocationArray[i][j];
        }
        }
        }
        static boolean checkSafeSystem(int processes[],
        int availableArray[],
        int maxArray[][],
        int allocationArray[][],
        int totalProcess,
        int totalResources)
        {
        int [][]needArray = new int[totalProcess][totalResources];
        findNeedValue(needArray, maxArray, allocationArray, totalProcess,
totalResources);
        boolean []finishProcesses = new boolean[totalProcess];
        int []safeSequenceArray = new int[totalProcess];
        int []workArray = new int[totalResources];
```

```
        for (int i = 0; i < totalResources ; i++) workArray[i] = availableArray[i];
        int counter = 0;
        while (counter < totalProcess)
        {
                boolean foundSafeSystem = false;
        for (int m = 0; m < totalProcess; m++)
        {
                if (finishProcesses[m] == false)
                {
                int j;
                for (j = 0; j < totalResources; j++)
                if (needArray[m][j] > workArray[j]) break;
                if (j == totalResources)
                { for (int k = 0 ; k < totalResources ; k++) workArray[k] +=
allocationArray[m][k];
                safeSequenceArray[counter++] = m;
                finishProcesses[m] = true;
                foundSafeSystem = true;
                }
                }
                }
                if (foundSafeSystem == false)
                {
                System.out.print("The system(20BCD7171) is not in the safe state
because lack of resources");
                return false;
                }
                }
                System.out.print("The system(20BCD7171) is in safe sequence and the
sequence is as follows: ");
                for (int i = 0; i < totalProcess ; i++)
                System.out.print("P"+safeSequenceArray[i] + " ");
                return true;
                }
                public static void main(String[] args)
                { int numberOfProcesses, numberOfResources; Scanner sc = new
Scanner(System.in);
                System.out.println("Enter total number of processes(20BCD7171)");
                numberOfProcesses = sc.nextInt();
                System.out.println("Enter total number of resources(20BCD7171)");
```

```
                numberOfResources = sc.nextInt();
                int processes[] = new int[numberOfProcesses];
                for(int i = 0; i < numberOfProcesses; i++)
                {
                processes[i] = i;
                }
                int availableArray[] = new int[numberOfResources];
                for( int i = 0; i < numberOfResources; i++)
                {
                System.out.println("Enter the availability of resource(20BCD7171)"+ i +":
");
                availableArray[i] = sc.nextInt();

                }
                int maxArray[][] = new int[numberOfProcesses][numberOfResources];
                for( int i = 0; i < numberOfProcesses; i++){
                for( int j = 0; j < numberOfResources; j++){
                        System.out.println("Enter the maximum resource(20BCD7171)"+ j
+" that can be allocated to process(20BCD7171)"+ i +": ");
                        maxArray[i][j] = sc.nextInt();
                        }
                        }
                        int allocationArray[][] = new
int[numberOfProcesses][numberOfResources];
                        for( int i = 0; i < numberOfProcesses; i++)
                        {
                        for( int j = 0; j < numberOfResources; j++)
                        {
                        System.out.println("How many instances of
resource(20BCD7171)"+ j +" are allocated to process(20BCD7171)"+ i +"? ");
                        allocationArray[i][j] = sc.nextInt();
                        }
                        }
                        checkSafeSystem(processes, availableArray, maxArray,
allocationArray, numberOfProcesses, numberOfResources);
                        }
                        }
```

OUTPUT:

```
Enter total number of processes(20BCD7171)
5
Enter total number of resources(20BCD7171)
3
Enter the availability of resource(20BCD7171)0:
10
Enter the availability of resource(20BCD7171)1:
5
Enter the availability of resource(20BCD7171)2:
7
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)0
:
7
Enter the maximum resource(20BCD7171)1 that can be allocated to process(20BCD7171)0
:
5
Enter the maximum resource(20BCD7171)2 that can be allocated to process(20BCD7171)0
:
3
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)1
:
3
Enter the maximum resource(20BCD7171)1 that can be allocated to process(20BCD7171)1
:
2
Enter the maximum resource(20BCD7171)2 that can be allocated to process(20BCD7171)1
:
2
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)2
:
9
Enter the maximum resource(20BCD7171)1 that can be allocated to process(20BCD7171)2
:
0
Enter the maximum resource(20BCD7171)2 that can be allocated to process(20BCD7171)2
:
2
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)3
:
2
Enter the maximum resource(20BCD7171)1 that can be allocated to process(20BCD7171)3
:
2
Enter the maximum resource(20BCD7171)2 that can be allocated to process(20BCD7171)3
:
2
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)4
:
4
```

```
Enter the maximum resource(20BCD7171)0 that can be allocated to process(20BCD7171)4
:
4
Enter the maximum resource(20BCD7171)1 that can be allocated to process(20BCD7171)4
:
3
Enter the maximum resource(20BCD7171)2 that can be allocated to process(20BCD7171)4
:
3
How many instances of resource(20BCD7171)0 are allocated to process(20BCD7171)0?
0
How many instances of resource(20BCD7171)1 are allocated to process(20BCD7171)0?
1
How many instances of resource(20BCD7171)2 are allocated to process(20BCD7171)0?
0
How many instances of resource(20BCD7171)0 are allocated to process(20BCD7171)1?
3
How many instances of resource(20BCD7171)1 are allocated to process(20BCD7171)1?
0
How many instances of resource(20BCD7171)2 are allocated to process(20BCD7171)1?
2
How many instances of resource(20BCD7171)0 are allocated to process(20BCD7171)2?
2
How many instances of resource(20BCD7171)1 are allocated to process(20BCD7171)2?
1
How many instances of resource(20BCD7171)2 are allocated to process(20BCD7171)2?
1
How many instances of resource(20BCD7171)0 are allocated to process(20BCD7171)3?
0
How many instances of resource(20BCD7171)1 are allocated to process(20BCD7171)3?
0
How many instances of resource(20BCD7171)2 are allocated to process(20BCD7171)3?
2
How many instances of resource(20BCD7171)0 are allocated to process(20BCD7171)4?
2
How many instances of resource(20BCD7171)1 are allocated to process(20BCD7171)4?
3
How many instances of resource(20BCD7171)2 are allocated to process(20BCD7171)4?
0
The system(20BCD7171) is in safe sequence and the sequence is as follows: P0 P1 P2
P3 P4

...Program finished with exit code 0
Press ENTER to exit console.
```