**Student Name:** Adithya Shankar L

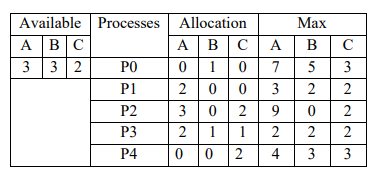
**Student ID:** 11803986 (B44)

**Email Address:** [adithyashankar10@gmail.com](mailto:adithyashankar10@gmail.com)

**GitHub Link:** https://github.com/ADI711/Bankers-algorithm-os.git

Problem:

**Ques. 19.** There are 5 processes and 3 resource types, resource A with 10 instances, B with 5 instances and C with 7 instances. Consider following and write a c code to find whether the system is in safe state or not?



**Ans:** We can solve this question by using an algorithm called as banker’s algorithm.

The Banker’s algorithm is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra.

Algorithm:

Resource Request Algorithm

This describes the behaviour of the system when a process makes a resource request in the form of a request matrix. The steps are:

1. If number of requested instances of each resource is less than the need (which was declared previously by the process), go to step 2.
2. If number of requested instances of each resource type is less than the available resources of each type, go to step 3. If not, the process has to wait because sufficient resources are not available yet.
3. Now, assume that the resources have been allocated. Accordingly do,

**Available = Available – Request(i)**

**Allocation(i) = Allocation(i) + Request(i)**

**Need(i) = Need(i) – Request(i)**

**Safety Algorithm**

1. Let Work and Finish be vectors of length **m** and **n**, respectively. Initially,

**Work = Available**

**Finish[i] = False for i=0, 1,…,n-1**

This means, initially, no process has finished and the number of available resources is represented by the **Available** array.

1. Find an index **i** such that both

**Finish[i] == False**

**Need[i] <= Work**

If there is no such i present, then proceed to step 4.

It means, we need to find an unfinished process whose need can be satisfied by the available resources. If no such process exists, just go to step 4.

1. Perform the following:

**Work = Work + Allocation**

**Finish[i] = True**

Go to step 2.

When an unfinished process is found, then the resources are allocated and the process is marked finished. And then, the loop is repeated to check the same for all other processes.

1. If **Finish[i] == true for all i**, then the system is in a safe state.

That means if all processes are finished, then the system is in safe state.

Description:

To implement the above algorithm first we implement the Resource Request algorithm so that we can know all the resources required for each process (max res.). Then we try to run the Safety algorithm and check whether the sequence is safe or not.

Code:

/\* Code done by Adithya Shankar (B44) \*/

// Banker's Algorithm

#include <stdio.h>

int main()

{

// P0, P1, P2, P3, P4 are the Process names

int a, b, i, j, k;

a = 5; // Number of processes

b = 3; // Number of resources

int all[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix

{ 2, 0, 0 }, // P1

{ 3, 0, 2 }, // P2

{ 2, 1, 1 }, // P3

{ 0, 0, 2 } }; // P4

int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix

{ 3, 2, 2 }, // P1

{ 9, 0, 2 }, // P2

{ 2, 2, 2 }, // P3

{ 4, 3, 3 } }; // P4

int avail[3] = { 3, 3, 2 }; // Available Resources

int n[a], ans[a], ind = 0;

for (k = 0; k < a; k++) {

n[k] = 0;

}

int need[a][b];

for (i = 0; i < a; i++) {

for (j = 0; j < b; j++)

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

}

int y = 0;

for (k = 0; k < 5; k++) {

for (i = 0; i < a; i++) {

if (n[i] == 0) {

int flag = 0;

for (j = 0; j < b; j++) {

if (need[i][j] > avail[j]){

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

for (y = 0; y < m; y++)

avail[y] += all[i][y];

n[i] = 1;

}

}

}

}

printf("Following is the SAFE State or Sequence\n");

printf("Sequence is : ")

for (i = 0; i < a - 1; i++)

printf(" P%d ->", ans[i]);

printf(" P%d", ans[a - 1]);

return (0);

}

Output:

