Name: Ngov Socheatdalin

ID: e20190679

Group: I4-GIC-B

Example:

Let us consider the following snapshot for understanding the banker's algorithm:

Processes	Allocation A B C	Max A B C	Available A B C
P0	1 1 2	4 3 3	2 1 0
P1	2 1 2	3 2 2	
P2	4 0 1	9 0 2	
P3	0 2 0	7 5 3	
P4	1 1 2	1 1 2	

1. calculate the content of the need matrix?
2. Check if the system is in a safe state?
3. Determine the total sum of each type of resource?

Solution:

1. The Content of the need matrix can be calculated by using the formula given below:

Need = Max – Allocation

Process	Need		
	A	B	C
P_0	3	2	1
P_1	1	1	0
P_2	5	0	1
P_3	7	3	3
P_4	0	0	0

2. Let us now check for the safe state.

Safe sequence:

1. For process P0, Need = (3, 2, 1) and

Available = (2, 1, 0)

Need <=Available = False

So, the system will move to the next process.

2. For Process P1, Need = (1, 1, 0)

Available = (2, 1, 0)

Need <= Available = True

Request of P1 is granted.

Available = Available +Allocation

= (2, 1, 0) + (2, 1, 2)

= (4, 2, 2) (New Available)

3. For Process P2, Need = (5, 0, 1)

Available = (4, 2, 2)

Need <=Available = False

So, the system will move to the next process.

4. For Process P3, Need = (7, 3, 3)

Available = (4, 2, 2)

Need <=Available = False

So, the system will move to the next process.

5. For Process P4, Need = (0, 0, 0)

Available = (4, 2, 2)

Need <= Available = True

Request of P4 is granted.

Available = Available + Allocation

= (4, 2, 2) + (1, 1, 2)

= (5, 3, 4) now, (New Available)

6. Now again check for Process P2, Need = (5, 0, 1)

Available = (5, 3, 4)

Need <= Available = True

Request of P2 is granted.

Available = Available + Allocation

= (5, 3, 4) + (4, 0, 1)

= (9, 3, 5) now, (New Available)

7. Now again check for Process P3, Need = (7, 3, 3)

Available = (9, 3, 5)

Need <=Available = True

The request for P3 is granted.

Available = Available +Allocation

= (9, 3, 5) + (0, 2, 0) = (9, 5, 5)

8. Now again check for Process P0, = Need (3, 2, 1)

= Available (9, 5, 5)

Need <= Available = True

So, the request will be granted to P0.

Safe sequence: < P1, P4, P2, P3, P0>

The system allocates all the needed resources to each process. So, we can say that the system is in a safe state.

3. The total amount of resources will be calculated by the following formula:

The total amount of resources= sum of columns of allocation + Available

= [8 5 7] + [2 1 0] = [10 6 7]

```c
//C program for Banker's Algorithm
#include <stdio.h>
int main()
{
    // P0, P1, P2, P3, P4 are the names of Process

    int n, r, i, j, k;
    n = 5; // Indicates the Number of processes
    r = 3; //Indicates the Number of resources
    int alloc[5][3] = { { 0, 0, 1 }, // P0 // This is Allocation Matrix
                        { 3, 0, 0 }, // P1
                        { 1, 0, 1 }, // P2
                        { 2, 3, 2 }, // P3
                        { 0, 0, 3 } }; // P4

    int max[5][3] = { { 7, 6, 3 }, // P0 // MAX Matrix
                      { 3, 2, 2 }, // P1
                      { 8, 0, 2 }, // P2
                      { 2, 1, 2 }, // P3
                      { 5, 2, 3 } }; // P4

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

    int f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++) {
        f[k] = 0;
    }
    int need[n][r];
    for (i = 0; i < n; i++) {
        for (j = 0; j < r; j++)
            need[i][j] = max[i][j] - alloc[i][j];
    }
    int y = 0;
    for (k = 0; k < 5; k++) {
        for (i = 0; i < n; i++) {
            if (f[i] == 0) {

                int flag = 0;
                for (j = 0; j < r; j++) {
                    if (need[i][j] > avail[j]){
                        flag = 1;
                        break;
                    }
                }

                if (flag == 0) {
                    ans[ind++] = i;
                    for (y = 0; y < r; y++)
                        avail[y] += alloc[i][y];
                    f[i] = 1;
                }
            }
        }
    }
```

```c
    printf("Th SAFE Sequence is as follows\n");
    for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
    printf(" P%d", ans[n - 1]);

    return (0);

}
```

```
Th SAFE Sequence is as follows
 P1 -> P3 -> P4 -> P0 -> P2
Process returned 0 (0x0)   execution time : 0.110 s
Press any key to continue.
```