Student Name: Md Asibur Rahman Akash
Student ID: 11800427
Section: K18PT
Email Address: text2mine@gmail.com
GitHub Link: https://github.com/Aakash-17

Question:24 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?

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

Description:

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Let **'n'** be the number of processes in the system and **'m'** be the number of resources types.

Available :

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available[i] = k means there are **'k'** instances of resource type $\mathbf{R_j}$

Max:

- It is a 2-d array of size '**n*m'** that defines the maximum demand of each process in a system.
- Max [i , j] = k means process $\mathbf{P_i}$ may request at most **'k'** instances of resource type $\mathbf{R_j}$.

Allocation:

- It is a 2-d array of size **'n*m'** that defines the number of resources of each type currently allocated to each process.
- Allocation [I , j] = k means process $\mathbf{P_i}$ is currently allocated **'k'** instances of resource type $\mathbf{R_j}$

Need :

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [i , j] = k means process $\mathbf{P_i}$ currently need **'k'** instances of resource type $\mathbf{R_j}$
  for its execution.

Code:-(Predefined values in the code

predefined_values of safe.c | user_input for safe.c

```c
/* Question 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?

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


#include <stdio.h>
int main()
{
    // P0, P1, P2, P3, P4 are the Process names here

    int n, m, i, j, k; ///declaration of variables
    n = 5; // Indicates the total number of processes of the system
    m = 3; // Indicates the total number of resources in the system
    int alloc[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix
                        { 2, 0, 0 }, // P1 // Indicates where the process you have received a resource
                        { 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 f[n], ans[n], ind = 0;
    for (k = 0; k < n; k++) {   //Sorting the process
        f[k] = 0;
    }
    int need[n][m];                 //Express how many more resources can be allocated in future
    for (i = 0; i < n; i++) {        //Sorting the process
        for (j = 0; j < m; j++)     //Sorting the process
            need[i][j] = max[i][j] - alloc[i][j];    //Need= maximum resources - currently allocated resources
    }
    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 < m; j++) {
                    if (need[i][j] > avail[j]){
                        flag = 1;
                        break;
                    }
                }

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

    printf("Following is the SAFE Sequence\n");
    for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);              //Sorting the process for safe state.
    printf(" P%d", ans[n - 1]);                 //Printing all hte process in safe state order

    return (0);

}
```

Compiler | Resources | Compile Log | Debug | Find Results

Output:

E:\LPU CLASS\Semester-4\CSE-316\MyosProject\predefined_values of safe.exe

```
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
--------------------------------
Process exited after 0.02084 seconds with return value 0
Press any key to continue . . .
```

Code:-(User is asked to enter the values)

predefined_values of safe.c | [*] user_input for safe.c

```c
/* Question 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?

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

#include<stdio.h>
int main()
{
    int num;
    int n;
    int i,j,k,c,c1;
    int avail[20],arr[10];
    int need[20][20],alloc[20][20],max[20][20];

    printf("\nEnter number of processes :");
    scanf("%d",&num);

    printf("\nEnter the number of resources available :");
    scanf("%d",&n);

    printf("\nEnter instances for resources(Press enter after entering each integer value) :\n");
    for(i=0;i<n;i++)
    {
        printf("R%d    ",i+1);
        scanf("%d",&avail[i]);
    }
    printf("\n Enter allocation matrix(INTEGER) with one space after each integer  \n");   //Allocation Matrix
    printf("\n       A B C \n");            //For pretty formatting output
    for(i=0;i<num;i++)
    {
        printf("p%d     ",    i);          arr[i]=0;       //to print the process number
        for(j=0;j<n;j++)
        {
            scanf("%d",    &alloc[i][j]);
        }
    }
    printf("\n Enter MAX matrix(INTEGER) with one space after each integer  \n");        //MAX Matrix
    printf("\n       A B C \n");           //For pretty formatting output
    for(i=0;i<num;i++)                                  //Sorting the process
    {
        printf("p%d     ",    i);                       //to print the process number
        for(j=0;j<n;j++)                                //Sorting the process
        {
            scanf("%d",&max[i][j]);
        }
    }
```

```c
    for(i=0;i<num;i++)                                  //Sorting the process
    {
        printf("\np%d\t",i) ;
        for(j=0;j<n;j++)                                //Sorting the process
        {
            need[i][j]=max[i][j]-alloc[i][j];           //Need= maximum resources - currently allocated resources
            printf("\t%d",need[i][j]);
        }
    }
    k=0;     c1=0;
    printf("\n\n");
    while(k<15)
    {
        for(i=0;i<num;i++)                              //Sorting the process
        {
            c=0;
            for(j=0;j<n;j++)                            //Sorting the process
            {
                if(arr[i]==1) break;
                if(need[i][j]<=avail[j])
                {
                    c++;
                }
                if(c==n)
                {
                    for(j=0;j<n;j++)
                    {
                        avail[j]+=alloc[i][j];
                    }
                    printf("p%d\t ->",i);  arr[i]=1;     c1++;
                }
            }
        }
        k++;
    }
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Line: 51 Col: 5 Sel: 0 Lines: 89 Length: 2805 Insert Press Ctrl+F11 to toggle fullscreen or Ctrl+F12 to toggle toolbars

Output:

E:\LPU CLASS\Semester-4\CSE-316\MyosProject\user_input for safe.exe

```
Enter number of processes :5

Enter the number of resources available :3

Enter instances for resources(Press enter after entering each integer value) :
R1    3 3 2
R2    R3
 Enter allocation matrix(INTEGER) with one space after each integer

       A B C
p0     0 1 0
p1     2 0 0
p2     3 0 2
p3     2 1 1
p4     0 0 2

 Enter MAX matrix(INTEGER) with one space after each integer

       A B C
p0     7 5 3
p1     3 2 2
p2     9 0 2
p3     2 2 2
p4     4 3 3

p0             7       4       3
p1             1       2       2
p2             6       0       0
p3             0       1       1
p4             4       3       1

p1       ->p3    ->p4    ->p0    ->p2    ->
--------------------------------
Process exited after 59.41 seconds with return value 5
Press any key to continue . . .
```

Question-19

There are 5 process and three (3) resource types, resource with A with 10 instance, B with 5 instance and C with 7 instance. Consider following and write a code to find whether the system in safe state or not?

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

Need Matrix:

Max - Allocation

	A	B	C
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

Safety Algorithm:

if

need $\leq$ available

the Execute Process

New available = Available + Allocatio

Else

do not execute go forward

P.0 $\rightarrow$ need > available

743 > 332

do not execute P0

P1 $\rightarrow$ need $\leq$ available

112 $\leq$ 332

execute P1

Now available = Available + Allocation

= 3 3 2 + 2 0 0

= 5 3 2

$P_2 \rightarrow$ need > available

$= 6\ 0\ 0 > 5\ 3\ 2$

Do not execute P2

$P_3 \rightarrow$ need $\leq$ available

$0\ 1\ 1 \leq 5\ 3\ 2$

Execute P3

Now available $= 5\ 3\ 2 + 2\ 1\ 1$

$= 7\ 4\ 3$

$P_4 \rightarrow$ need $\leq$ available

$4\ 3\ 1 \leq 7\ 4\ 3$

Execute P4

Now available $= 7\ 4\ 3 + 0\ 0\ 2$

$= 7\ 4\ 5$

P0 → need ≤ available

= 743 ≤ 745

Now available = 745 + 010

= (7,5,5)

P2 → need ≤ available

= 600 ≤ 7 5 5

Execute P2

New

Availabe

= 755 + 302

= (10, 5, 7)
↓ ↓ ↓
A B C

{ It is in Safe state }

SAFE SEQUENCE:

⟨ P1 , P3 , P4 , P0 , P2 ⟩