

Registration Number – 21BAI1300

Name – Roshan Naidu

Dr. Leki Chom Thungon – L15 + L16 – BCSE303P

Operating Systems Lab FAT

1. Banker's Algorithm – To check if a safe sequence or not by taking inputs given in the FAT Lab test paper.

Code-

```
#include <stdio.h>

int main()
{
    int i, j, n, r, alloc[30][30], avail[10], remneed[30][30], max[30][30], work[30], w[30], y;

    int sum, ind = 0;

    int flag = 0;

    int safeseq[30];

    // To get the total number of processes incoming :-

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter the number of resources present: ");
    scanf("%d", &r);

    printf("Enter allocation matrix:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < r; j++)
        {
            scanf("%d", &alloc[i][j]);
        }
    }
}
```

```
}
```

```
printf("Enter maximum of the allocation matrix: \n");
```

```
for (i = 0; i < n; i++)  
{  
    for (j = 0; j < r; j++)  
    {  
        scanf("%d", &max[i][j]);  
    }  
}
```

```
printf("Enter the available resources: \n");
```

```
for (i = 0; i < r; i++)  
{  
    scanf("%d", &avail[i]);  
}
```

```
// Calculate need matrix :-
```

```
for (i = 0; i < n; i++)  
{  
    for (j = 0; j < r; j++)  
    {  
        remneed[i][j] = max[i][j] - alloc[i][j];  
    }  
}
```

```
// Initialize work and finish arrays
```

```
for (i = 0; i < r; i++)  
{  
    work[i] = avail[i];  
}
```

```
int finish[n];
```

```
for (i = 0; i < n; i++)  
{  
    finish[i] = 0;  
}
```

```
// Applying the Banker's Algorithm
```

```
int k = 0;
```

```
// Initialize the index for safe sequence array
```

```
int found;
```

```
while (k < n)
```

```
{
```

```
    found = 0;
```

```
    for (i = 0; i < n; i++)
```

```
    {
```

```
        if (finish[i] == 0)
```

```
        {
```

```
            flag = 0;
```

```
            for (j = 0; j < r; j++)
```

```
            {
```

```
                if (remneed[i][j] > work[j])
```

```
                {
```

```
                    flag = 1;
```

```
                    break;
```

```
                }
```

```
            }
```

```
        if (flag == 0)
```

```
        {
```

```
            w[k] = i;
```

```
            k++;
```

```
            for (y = 0; y < r; y++)
```

```
                work[y] += alloc[i][y];
```

```
                finish[i] = 1;
```

```
                found = 1;
```

```
        }
```

```
    }
```

```
}
```

```
    if (found == 0)
```

```
        break;
```

```
}
```

```
if (k == n)
```

```
// All processes were safely executed
```

```
{
```

```
    printf("Safe Sequence: ");
```

```
    for (i = 0; i < n; i++)
```

```
    {
```

```
        printf("%d ", w[i]);
```

```
        safeseq[i] = w[i];
```

```

    }
    printf("\n");
}

else

// There is no safe sequence
{
    printf("\nThere is no safe sequence.\n");
}

return 0;
}

```

OUTPUT-

The screenshot shows a code editor with the source code of a C program and a terminal window displaying its execution. The code implements the Banker's Algorithm to check for a safe sequence of processes based on their maximum resource requirements, current allocations, and available resources.

Code Editor (bank.c):

```

46 for (i = 0; i < r; i++)
47 {
48     scanf("%d", &avail[i]);
49 }
50
51 // Calculate need matrix :-
52
53 for (i = 0; i < n; i++)
54 {
55     for (j = 0; j < r; j++)
56     {
57         remneed[i][j] = max[i][j] - alloc[i][j];
58     }
59 }
60
61
62 // Initialize work and finish arrays
63
64 for (i = 0; i < r; i++)
65 {
66     work[i] = avail[i];
67 }
68
69 int finish[n];
70
71 for (i = 0; i < n; i++)
72 {
73     finish[i] = 0;
74 }
75
76 // Applying the Banker's Algorithm
77 int k = 0; // Initialize the index for safe sequence array
78
79 int found;
80
81 while (k < n)
82 {
83     found = 0;
84     for (i = 0; i < n; i++)
85     {
86         if (finish[i] == 0)
87         {
88             flag = 0;
89             for (j = 0; j < r; j++)
90             {

```

Terminal Window (ex2@AB1205BSC509: ~/Desktop):

```

File Edit View Search Terminal Help
Enter maximum of the allocation matrix:
Enter the available resources:

There is no safe sequence.
ex2@AB1205BSC509:~/Desktop$ gcc bank.c
ex2@AB1205BSC509:~/Desktop$ ./a.out
Enter number of processes: 4
Enter the number of resources present: 3
Enter allocation matrix:
4
5
6
4
3
3
4
1
4
3
3
2
Enter maximum of the allocation matrix:
2
1
5
2
1
2
4
1
3
3
3
2
Enter the available resources:
2
1
1
Safe Sequence: 0 1 2 3

```

1. Shell Programming – to print if 3 numbers are equal by taking input from the terminal command window.

Code-

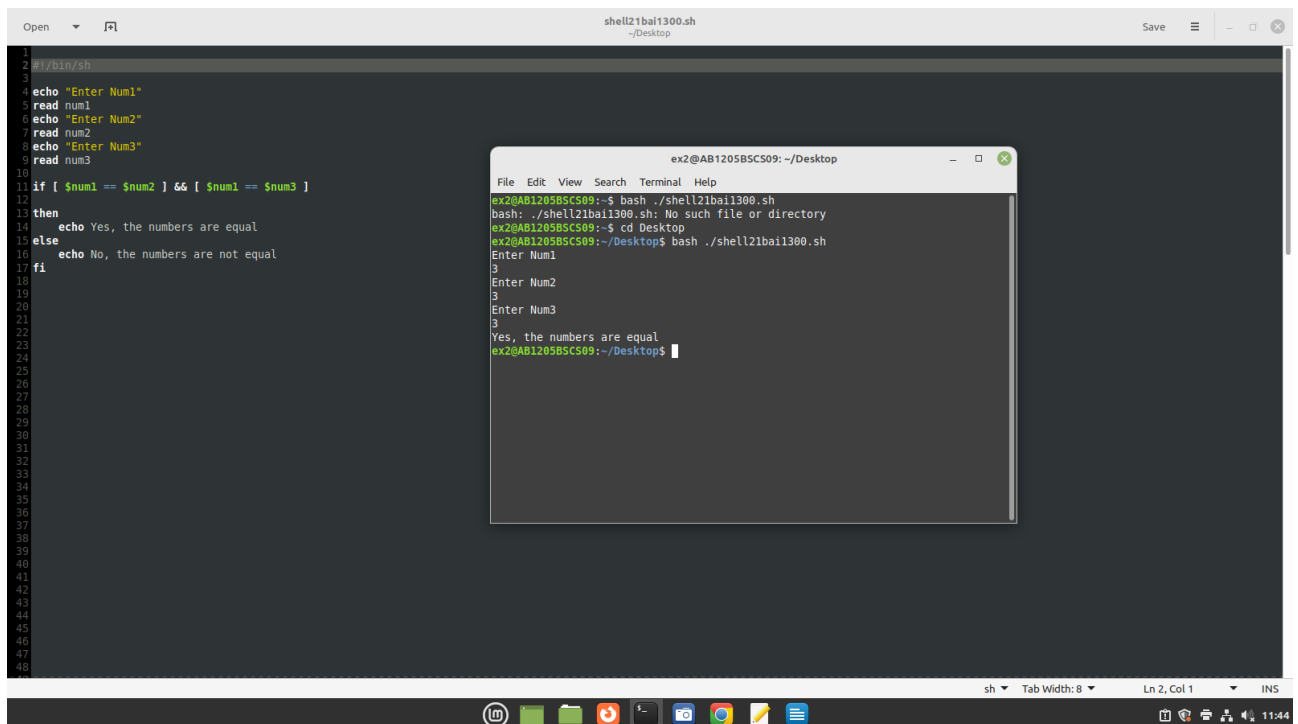
```
#!/bin/sh

echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3

if [ $num1 == $num2 ] && [ $num1 == $num3 ]

then
    echo Yes, the numbers are equal
else
    echo No, the numbers are not equal
fi
```

OUTPUT-



```
1 #!/bin/sh
2
3
4 echo "Enter Num1"
5 read num1
6 echo "Enter Num2"
7 read num2
8 echo "Enter Num3"
9 read num3
10
11 if [ $num1 == $num2 ] && [ $num1 == $num3 ]
12 then
13     echo Yes, the numbers are equal
14 else
15     echo No, the numbers are not equal
16 fi
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

ex2@AB1205BSC509: ~/Desktop

```
File Edit View Search Terminal Help
ex2@AB1205BSC509:~$ bash ./shell21bai1300.sh
bash: ./shell21bai1300.sh: No such file or directory
ex2@AB1205BSC509:~$ cd Desktop
ex2@AB1205BSC509:~/Desktop$ bash ./shell21bai1300.sh
Enter Num1
3
Enter Num2
3
Enter Num3
3
Yes, the numbers are equal
ex2@AB1205BSC509:~/Desktop$
```

sh Tab Width: 8 Ln 2, Col 1 INS 11:44