

Student Name-Sahil shukla

Student ID: 11804189

Email Address:-iamsahilshukla@gmail.com

GitHub Link: <https://github.com/iamsahilshukla/cse316-os-prjoect.git>

complexity:- $O(n+nm)=O(n*m)$

Algo-Brute Force

Question:24 Consider following and Generate a solution in C to find whether the system is in safe state or not?

Available				Processes	Allocation				Max			
A	B	C	D		A	B	C	D	A	B	C	D
1	5	2	0	P0	0	0	1	2	0	0	1	2
				P1	1	0	0	0	1	7	5	0
				P2	1	3	5	4	2	3	5	6
				P3	0	6	3	2	0	6	5	2
				P4	0	0	1	4	0	6	5	6

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 **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 **P_i** may request at most '**k**' instances of resource type **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 **P_i** is currently allocated '**k**' instances of resource type **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 **P_i** currently need '**k**' instances of resource type **R_j**

for its execution.

- $\text{Need}[i, j] = \text{Max}[i, j] - \text{Allocation}[i, j]$

Code:-(Predefined values in the code)

```
1  /* Question 23-- Consider following and Generate a solution in C to find whether the system is in safe state or not
2
3
4  Available      Processes      Allocation      Max
5  A  B  C  D      P0      A  B  C  D      A  B  C  D
6  1  5  2  0      P1      0  0  1  2      0  0  1  2
7      P2      1  0  0  0      1  7  5  0
8      P3      1  3  5  4      2  3  5  6
9      P4      0  6  3  2      0  6  5  2
10     P4      0  0  1  4      0  6  5  6
11
12 #include <stdio.h>
13 int main()
14 {
15     // P0, P1, P2, P3, P4 are the Process names here
16
17     int n, m, i, j, k; // declaration of variables
18     n = 5; // Indicates the total number of processes of the system
19     m = 4; // Indicates the total number of resources in the system
20     int alloc[5][4] = { { 0, 0, 1, 2 }, // P0 // Allocation Matrix
21                         { 1, 0, 0, 0 }, // P1 // Indicates where the process you have received a resource
22                         { 1, 3, 5, 4 }, // P2
23                         { 0, 6, 3, 2 }, // P3
24                         { 0, 0, 1, 4 } }; // P4
25
26     int max[5][4] = { { 0, 0, 1, 2 }, // P0 // MAX Matrix
27                     { 1, 7, 5, 0 }, // P1
28                     { 2, 3, 5, 6 }, // P2
29                     { 0, 6, 5, 2 }, // P3
30                     { 0, 6, 5, 6 } }; // P4
31 }
```

```

32 int avail[4] = { 1, 5, 2, 0 }; // Available Resources
33
34 int f[n], ans[n], ind = 0;
35 for (k = 0; k < n; k++) { //Sorting the process
36     f[k] = 0;
37 }
38 int need[n][m]; //Express how many more resources can be allocated in future
39 for (i = 0; i < n; i++) { //Sorting the process
40     for (j = 0; j < m; j++) //Sorting the process
41         need[i][j] = max[i][j] - alloc[i][j]; //Need= maximum resources - currently allocated resources
42 }
43 int y = 0;
44 for (k = 0; k < 5; k++) {
45     for (i = 0; i < n; i++) {
46         if (f[i] == 0) {
47
48             int flag = 0;
49             for (j = 0; j < m; j++) {
50                 if (need[i][j] > avail[j]){
51                     flag = 1;
52                     break;
53                 }
54             }
55
56             if (flag == 0) {
57                 ans[ind++] = i;
58                 for (y = 0; y < m; y++)
59                     avail[y] += alloc[i][y];
60                 f[i] = 1;
61             }

```

```

62 |         }
63 |     }
64 | }
65 |
66 | printf("Following is the SAFE Sequence\n");
67 | for (i = 0; i < n - 1; i++) //Sorting the process for safe state.
68 |     printf(" P%d ->", ans[i]); //Printing all hte process in safe state order
69 |     printf(" P%d", ans[n - 1]);
70 |
71 | return (0);
72 |
73 | }
74 |

```

```

C:\Users\bhave\Documents\ca2.exe
Following is the SAFE Sequence
P0 -> P2 -> P3 -> P4 -> P1
-----
Process exited after 0.0208 seconds with return value 0
Press any key to continue . . .

```

Output:-

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


```

65 {
66     for(i=0;i<num;i++) //Sorting the process
67     {
68         c=0;
69         for(j=0;j<n;j++) //Sorting the process
70         {
71             if(arr[i]==1) break;
72             if(need[i][j]<=avail[j])
73             {
74                 c++;
75             }
76             if(c==n)
77             {
78                 for(j=0;j<n;j++)
79                 {
80                     avail[j]+=alloc[i][j];
81                 }
82                 printf("p%d\t->",i); arr[i]=1; c1++;
83             }
84         }
85     }
86     k++;
87 }
88 }

33 printf("\n Enter allocation matrix(INTEGER) with one space after each integer \n"); //Allocation Matrix
34 printf("\n      A B C D \n"); //For pretty formatting output
35 for(i=0;i<num;i++)
36 {
37     printf("p%d", i); arr[i]=0; //to print the process number
38     for(j=0;j<n;j++)
39     {
40         scanf("%d", &alloc[i][j]);
41     }
42 }
43 printf("\n Enter MAX matrix(INTEGER) with one space after each integer \n"); //MAX Matrix
44 printf("\n      A B C D \n"); //For pretty formatting output
45 for(i=0;i<num;i++) //Sorting the process
46 {
47     printf("p%d", i); //to print the process number
48     for(j=0;j<n;j++) //Sorting the process
49     {
50         scanf("%d",&max[i][j]);
51     }
52 }
53 for(i=0;i<num;i++) //Sorting the process
54 {
55     printf("\np%d\t",i) ;
56     for(j=0;j<n;j++) //Sorting the process
57     {
58         need[i][j]=max[i][j]-alloc[i][j]; //Need= maximum resources - currently allocated resources
59         printf("\t%d",need[i][j]);
60     }
61 }
62 k=0; c1=0;
63 printf("\n\n");
64 while(k<15)

```


1 /* Question 24-- Consider following and Generate a solution in C to find whether the system is in safe state or not?

Available				Processes	Allocation				Max			
A	B	C	D		A	B	C	D	A	B	C	D
1	5	2	0	P0	0	0	1	2	0	0	1	2
				P1	1	0	0	0	1	7	5	0
				P2	1	3	5	4	2	3	5	6
				P3	0	6	3	2	0	6	5	2
				P4	0	0	1	4	0	6	5	6

12 #include<stdio.h>

13 int main()

14 {

15 int num;

16 int n;

17 int i,j,k,c,c1;

18 int avail[20],arr[10];

19 int need[20][20],alloc[20][20],max[20][20];

20

21 printf("\nEnter number of processes :");

22 scanf("%d",&num);

23

24 printf("\nEnter the number of resources available :");

25 scanf("%d",&n);

26

27 printf("\nEnter instances for resources(Press enter after entering each integer value) :\n");

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

29 {

30 printf("R%d ",i+1);

31 scanf("%d",&avail[i]);

32 }

C:\Users\bhave\Desktop\CA\user_input.exe

Enter number of processes :5

Enter the number of resources available :4

Enter instances for resources(Press enter after entering each integer value) :

R1 1
R2 5
R3 2
R4 0

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

	A	B	C	D
p0	0	0	1	2
p1	1	0	0	0
p2	1	3	5	4
p3	0	6	3	2
p4	0	0	1	4

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

	A	B	C	D
p0	0	0	1	2
p1	1	7	5	0
p2	2	3	5	6
p3	0	6	5	2
p4	0	6	5	6

p0		0	0	0	0
p1		0	7	5	0
p2		1	0	0	2
p3		0	0	2	0
p4		0	6	4	2

p0 ->p2 ->p3 ->p4 ->p1 ->

Process exited after 56.9 seconds with return value 5
Press any key to continue . . .

Q3. Write a multithreaded program in C that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

Description:-

According to question the user will enter a number in the command line then the program will create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

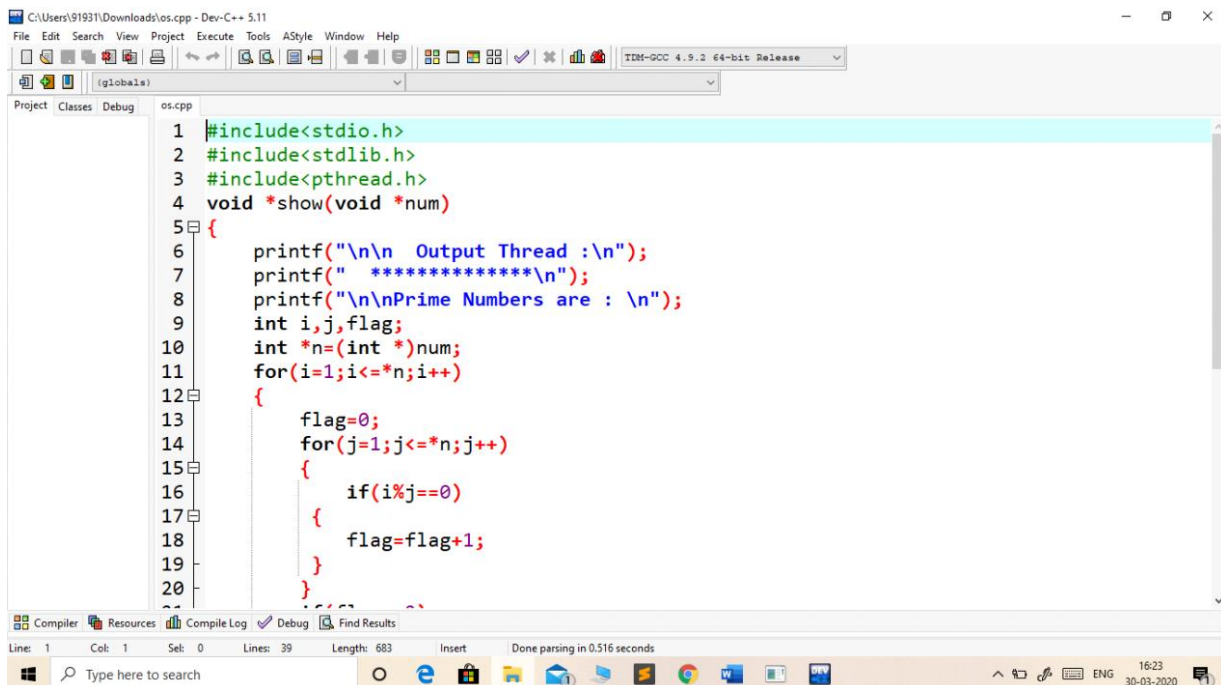
After compiling the code when we will run the code that according to the number entered it will create separate thread that will give us all the prime numbers less than or equal to the number entered by us.

Algorithm:-

Rough Logic

Complexity:- $O(n^3)$

Code-

The image is a screenshot of a C++ IDE, likely Dev-C++, showing a source code file named 'os.cpp'. The code is a multithreaded program designed to find and output prime numbers up to a user-defined limit. It includes standard headers for input/output, standard library, and pthreads. A function 'show' is defined, which takes a pointer to an integer 'num' and prints the output thread's start, followed by the prime numbers. The main function calls 'show' with the user input. The code uses a nested loop to check for divisibility, which results in a time complexity of O(n^3). The IDE interface includes a menu bar, a toolbar, a project explorer on the left, and a status bar at the bottom showing the current line and column numbers.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<pthread.h>
4 void *show(void *num)
5 {
6     printf("\n\n Output Thread : \n");
7     printf(" ***** \n");
8     printf("\n\nPrime Numbers are : \n");
9     int i,j,flag;
10    int *n=(int *)num;
11    for(i=1;i<=*n;i++)
12    {
13        flag=0;
14        for(j=1;j<=*n;j++)
15        {
16            if(i%j==0)
17            {
18                flag=flag+1;
19            }
20        }
21    }
```

C:\Users\91931\Downloads\os.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(global)

Project Classes Debug os.cpp

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<pthread.h>
4  void *show(void *num)
5  {
6      printf("\n\n Output Thread :\n");
7      printf(" *****\n");
8      printf("\n\nPrime Numbers are : \n");
9      int i,j,flag;
10     int *n=(int *)num;
11     for(i=1;i<=*n;i++)
12     {
13         flag=0;
14         for(j=1;j<=*n;j++)
15         {
16             if(i%j==0)
17             {
18                 flag=flag+1;
19             }
20         }
21     }
22 }

```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 39 Length: 683 Insert Done parsing in 0.516 seconds

Type here to search

ENG 16:23 30-03-2020

```
C:\Users\91931\Downloads\os1.exe
```

```
Input Thread :  
*****  
  
Enter a number upto which prime numbers are required:-  
52  
  
Output Thread :  
*****  
  
Prime Numbers are :  
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47  
-----  
Process exited after 5.904 seconds with return value 0  
Press any key to continue . . .
```

C:\Users\91931\Downloads\os.exe

Input Thread :

Enter a number upto which prime numbers are required:-
122

Output Thread :

Prime Numbers are :
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113

Process exited after 19.98 seconds with return value 0
Press any key to continue . . .



Type here to search

