

10. Consider the following process table with number of processes that contains allocation field (for showing the number of resources of type: A, B and C allocated to each process in the table), max field (for showing the maximum number of resources of type: A, B, and C that can be allocated to each process). Write a program to calculate the entries of need matrix using the formula: $(\text{Need})_i = (\text{Max})_i - (\text{Allocation})_i$

Process	Allocation	Max	Available
	A B C	A B C	A B C
P0	1 1 2	5 4 4	3 2 1
P1	2 1 2	4 3 3	
P2	3 0 1	9 1 3	

3

P3	0 2 0	8 6 4	
P4	1 1 2	2 2 3	

```
1  #include <stdio.h>
2  int main()
3  {
4
5      int n, m, i, j, k;
6      n = 5;
7      m = 3;
8      int alloc[5][3] = {{1, 1, 2},
9                          {2, 1, 2},
10                         {3, 0, 1},
11                         {0, 2, 0},
12                         {1, 1, 2}};
13
14
15      int max[5][3] = {{5, 4, 4},
16                       {4, 3, 3},
17                       {9, 1, 3},
18                       {8, 6, 4},
19                       {2, 2, 3}};
20
21      int avail[3] = {3, 2, 1};
22
23      int f[n], ans[n], ind = 0;
24      for (k = 0; k < n; k++)
25      {
26          f[k] = 0;
27      }
28      int need[n][m];
29      for (i = 0; i < n; i++)
30      {
```

```
31     for (j = 0; j < m; j++)
32         need[i][j] = max[i][j] - alloc[i][j];
33     }
34     int y = 0;
35     for (k = 0; k < 5; k++)
36     {
37         for (i = 0; i < n; i++)
38         {
39             if (f[i] == 0)
40             {
41                 int flag = 0;
42                 for (j = 0; j < m; j++)
43                 {
44                     if (need[i][j] > avail[j])
45                     {
46                         flag = 1;
47                         break;
48                     }
49                 }
50                 if (flag == 0)
51                 {
52                     ans[ind++] = i;
53                     for (y = 0; y < m; y++)
54                         avail[y] += alloc[i][y];
55                     f[i] = 1;
56                 }
57             }
58         }
59     }
60     int flag = 1;
```

```
49     }
50     if (flag == 0)
51     {
52         ans[ind++] = i;
53         for (y = 0; y < m; y++)
54             avail[y] += alloc[i][y];
55         f[i] = 1;
56     }
57 }
58 }
59 }
60 int flag = 1;
61 for (int i = 0; i < n; i++)
62 {
63     if (f[i] == 0)
64     {
65         flag = 0;
66         printf("The following system is not safe");
67         break;
68     }
69 }
70 if (flag == 1)
71 {
72     printf("Following is the SAFE Sequence\n");
73     for (i = 0; i < n - 1; i++)
74         printf(" P%d ->", ans[i]);
75     printf(" P%d", ans[n - 1]);
76 }
77 return (0);
78 }
```

C:\Users\hp\Documents\bankers algorithm 10.exe

Following is the SAFE Sequence

P1 -> P4 -> P0 -> P2 -> P3

Process exited after 0.03684 seconds with return value 0

Press any key to continue . . .

Type here to search



33°C Partly sunny



13:48
04-05-2023