

Lab Assignment-6

Name: Sadman Sharif

ID: 20101107

Section: 02

Course: CSE321

Submission Date: 15/12/2022

Task-1

```
#include <stdio.h>
int main()
{
    int n, m, i, j, k;
    n = 5; // Number of processes
    m = 4; // Number of resources

    int alloc[5][4] = { { 0, 1, 0, 3 }, // P0    // Allocation Matrix
                        { 2, 0, 0, 0 }, // P1
                        { 3, 0, 2, 0 }, // P2
                        { 2, 1, 1, 5 }, // P3
                        { 0, 0, 2, 2 } }; // P4

    int max[5][4] = { { 6, 4, 3, 4 }, // P0 // MAX Matrix
                     { 3, 2, 2, 1 }, // P1
                     { 9, 1, 2, 6 }, // P2
                     { 2, 2, 2, 8 }, // P3
                     { 4, 3, 3, 7 } }; // P4

    int avail[4] = {3,3,2,1}; // Available Resources

    int f[n];
    int result[n];
    int ind=0;

    for (k=0;k<n; k++) {
        f[k]=0;
    }

    int need[n][m];

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++){
            need[i][j]=max[i][j]-alloc[i][j];}    // Calculated NEED
        }

    int y = 0;
```

```

for (k = 0; k < 5; k++) {
    for (i=0; i<n;i++) {
        if (f[i]==0) {

            int temp=0;

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

            if (temp==0) {
                result[ind++] = i;

                for (y = 0; y < m; y++)
                    avail[y]=avail[y]+alloc[i][y];
                f[i] = 1;
            }
        }
    }
}

int temp=1;

for(int i=0;i<n;i++){

    if(f[i]==0)
    {
        temp=0;
        printf("DEADLOCK AHEAD!");
        break;
    }
}

if(temp==1){

























    printf("SAFE HERE!:\n");

}




return (0);
}

```

Task-1 (Output)



main.c



```
1 #include <stdio.h>
2 int main()
3 {
4
5     int n, m, i, j, k;
6     n = 5; // Number of processes
7     m = 4; // Number of resources
8
9     int alloc[5][4] = { { 0, 1, 0, 3 }, // P0    // Allocation Matrix
10       { 2, 0, 0, 0 }, // P1
11       { 3, 0, 2, 0 }, // P2
12       { 2, 1, 1, 5 }, // P3
13       { 0, 0, 2, 2 } }; // P4
14
15     int max[5][4] = { { 6, 4, 3, 4 }, // P0 // MAX Matrix
16       { 3, 2, 2, 1 }, // P1
17       { 9, 1, 2, 6 }, // P2
18       { 2, 2, 2, 8 }, // P3
19       { 4, 3, 3, 7 } }; // P4
20
21
22     int avail[4] = {3,3,2,1}; // Available Resources
23
24     int f[n];
25     int result[n];
26     int ind=0;
27
28     for (k=0;k<n; k++) {
29         f[k]=0;
30     }
31
32     int need[n][m];
33
34     for (i = 0; i < n; i++) {
35         for (j = 0; j < m; j++)
36             need[i][j]=max[i][j]-alloc[i][j];    // Calculated NEED
37     }
```

Output

/tmp/3ZIdBpGT1P.o
DEADLOCK AHEAD!

Task-2

```
#include <stdio.h>

int main()
{
    int n, m, i, j, k;
    n = 6; // Number of processes
    m = 4; // Number of resources

    int alloc[6][4] = { { 0, 1, 0, 3 }, // P0 // Allocation Matrix
                        { 2, 0, 0, 3 }, // P1
                        { 3, 0, 2, 0 }, // P2
                        { 2, 1, 1, 5 }, // P3
                        { 0, 0, 2, 2 }, // P4
                        { 1, 2, 3, 1 } }; //P5

    int max[6][4] = { { 6, 4, 3, 4 }, // P0 // MAX Matrix
                     { 3, 2, 2, 4 }, // P1
                     { 9, 1, 2, 6 }, // P2
                     { 2, 2, 2, 8 }, // P3
                     { 4, 3, 3, 7 }, // P4
                     { 6, 2, 6, 5 } }; //P5

    int avail[4] = {2,2,2,1};

    int f[n];
    int result[n];
    int ind=0;
```

```

for (k=0;k<n; k++) {
    f[k]=0;
}

int need[n][m];

for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
        need[i][j]=max[i][j]-alloc[i][j];    // Calculated NEED
}

int y = 0;
for (k = 0; k < 5; k++) {
    for (i=0; i<n;i++) {
        if (f[i]==0) {
            int temp=0;
            for (j=0; j<m;j++) {
                if (need[i][j]>avail[j]){
                    temp=1;
                    break;
                }
            }

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

```

```

        }
    }
}
int temp=1;

for(int i=0;i<n;i++){

    if(f[i]==0)
    {
        temp=0;
        printf("Not a safe sequence");
        break;
    }
}

if(temp==1){

    printf("Safe Sequence:\n");









    for (i=0;i<n-1;i++)
        printf("P%d -->",result[i]);
    printf(" P%d", result[n-1]);

}



return (0);
}

```

Task-2 (Output)



main.c



Run

```
1  #include <stdio.h>
2  int main()
3  {
4
5      int n, m, i, j, k;
6      n = 6; // Number of processes
7      m = 4; // Number of resources
8
9
10     int alloc[6][4] = { { 0, 1, 0, 3 }, // P0  // Allocation Matrix
11                          { 2, 0, 0, 3 }, // P1
12                          { 3, 0, 2, 0 }, // P2
13                          { 2, 1, 1, 5 }, // P3
14                          { 0, 0, 2, 2 }, // P4
15                          { 1, 2, 3, 1 } }; //P5
16
17     int max[6][4] = { { 6, 4, 3, 4 }, // P0  // MAX Matrix
18                      { 3, 2, 2, 4 }, // P1
19                      { 9, 1, 2, 6 }, // P2
20                      { 2, 2, 2, 8 }, // P3
21                      { 4, 3, 3, 7 }, // P4
22                      { 6, 2, 6, 5 } }; //P5
23
24
25
26     int avail[4] = {2,2,2,1};
27
28     int f[n];
29     int result[n];
30     int ind=0;
31
32     for (k=0;k<n; k++) {
33         f[k]=0;
34     }
35
36     int need[n][m];
37
```

Output

```
/tmp/G2cWR79LfN.o
Safe Sequence:
P1 -->P3 -->P4 -->P5 -->P0 --> P2
```