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

```c
#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
    }
```

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

```c
#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];

```

Output

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