Fall 2022
CSE 321 Operating Systems
Lab Assignment 6
Total Marks: 20

Question 1 {10 Marks}

Write a program in c to detect if the system will face any deadlock in the future. If a deadlock is detected then print "**DEADLOCK AHEAD!**" otherwise print "**SAFE HERE!**". The situation is given below. (Allowed to use Banker's Algorithm).

Note: The code can be implemented in several different ways. Make sure your code works for any input but submit the code with following parameters and add the screenshot of the outputs in the doc.

```
int n = 5; // Number of processes
int 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 total[4] = {10, 5, 7, 11};      //Total resources
int avail[4];      //Available resources [need to calculate]
```

Output:

```
DEADLOCK AHEAD!
```

Question 2 {10 Marks}

Write a c program that will generate the safe sequence of process execution for the situation given below:(Use Banker's Algorithm).

Note: The code can be implemented in several different ways. Make sure your code works for any input but submit the code with following parameters and add the screenshot of the outputs in the doc.

```
int n = 6; // Number of processes
int 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 total[4] = { 10, 6, 10, 15};   //total resource

int avail[4];      //Available resources [need to calculate]
```

Output:

```
Safe Sequence:
P2 --> P4 --> P5 --> P6 --> P1 --> P3
```