

Lab 02 - Classes & Objects

Instructions:

- The board of a 4×4 Tic-Tac-Toe game can be represented by a string with a size of 16. Furthermore, the tokens of players are represented with the characters 'O' and 'X' and blank spaces are represented by the character '*'. For instance, an example of an empty board is represented on the left

```
      0  1  2  3
0  *  *  *  *
1  *  *  *  *
2  *  *  *  *
3  *  *  *  *
```

Empty Board Display

```
      0  1  2  3
0  0  1  2  3
1  4  5  6  7
2  8  9 10 11
3 12 13 14 15
```

String Indices Correlation

where the correlation of the indices of the elements of the string is listed on the right. Your objective is to create a *Board* class and define a couple of functions that operate on *Board* objects.

- Your submissions must be submitted to the GitHub repository in the Lab02 directory.
- Cheating of any kind is prohibited and will not be tolerated.
- Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the lab.

Grading

Task	Name	Maximum Points	Points Earned
1		2	
2		1.5	
3		1.5	
Total		5	

Task 1

In a header file named "T4Board.h" within the namespace *LB2* define a class named *Board* that must contain

- a public string field named *grid*.
- a public int field named *currentPlayer*.
- a public static constant string field named *tokens* that is initialized to "OX".
- a public default constructor that assigns a string of 16 asterisks to *grid* and assigns 0 to *currentPlayer*.
- a public copy constructor.
- a public assignment operator.
- a public empty destructor.

Task 2

In a header file named "T4Win.h" within the namespace *LB2* define a bool function named *Won()* that takes a constant *Board* reference parameter. It returns true only if four non-blank characters are in a row horizontally, vertically, or diagonally of the *grid* field of the parameter; otherwise, it returns false.

Task 3

In a header file named "T4Display.h" within the namespace *LB2* define a void function named *Display()* that takes a constant *Board* reference parameter. It displays the *grid* field of the parameter in the same format as the illustration in the instruction [i.e it includes the row-column indices and has spaces between values].

Extra Credit

In a cpp file, define an int function named *UniquePrimeFactors()* whose header is

```
int UniquePrimeFactors(unsigned int n)
```

that returns the number of unique prime factors of *n*. For instance, *UniquePrimeFactors(40)* and *UniquePrimeFactors(71)* will return 2 and 1 respectively.

Hint: If a number is composite, its prime factors are at most its square root (2 points)