Problem Set 07 - Encapsulation

Complete each task below. Remember to include all header files in the accompanying cpp file. Use `isalpha()`, `toupper()`, and `tolower()` from the *cctype* library, use `setfill()` and `setw()` from the *iomanip* library when applicable.

Tasks:

1. Create a header file named `"Age.h"` and define the class *Age* within the namespace *PS7*. The class must contain
   - ☐ a private int field named *value*.
   - ☐ a public default constructor that assigns 1 to *value*.
   - ☐ a public copy constructor.
   - ☐ a public assignment operator.
   - ☐ a public empty destructor.
   - ☐ a public void setter method for *value* that assigns the parameter to *value* only if it is positive.
   - ☐ a public constant getter method for *value*.
   - ☐ a public string constant method named `ToString()` that takes no parameters. It returns a string in the format

     x `year(s)` y `day(s)`

     where x is the integer quotient of *value* divided by 365 and y is the integer remainder of *value* divided by 365.
   - ☐ an ostream operator that returns an outcome in the same format as `ToString()`.

   Afterward, within the accompanying cpp file, within the main function, declare a *Age* object. Try to assign the field of the object a negative value, and then, display the object. Next, assign the field of the object a positive value, and then, display the object.
2. Create a header file named `"Word.h"` and define the class *Word* within the namespace *PS7*. The class must contain
   - ☐ a private string field named *value*.
   - ☐ a public default constructor that assigns `"word"` to *value*.
   - ☐ a public copy constructor.
   - ☐ a public assignment operator.
   - ☐ a public empty destructor.
   - ☐ a public void setter method for *value* that assigns the parameter to *value* only if it contains only letters.
   - ☐ a public constant getter method for *value*.
   - ☐ a public string constant method named `ToString()` that takes no parameters. It returns the value of *value* in capitalized format [the first letter is uppercase while the remaining letters are lowercase].
   - ☐ an ostream operator that returns an outcome in the same format as `ToString()`.

   Afterward, within the accompanying cpp file, within the main function, declare a *Word* object. Try to assign the field of the object a string that contains at least one character that is not a letter, and then, display the object. Next, assign the field of the object a string that contains only letters, and then, display the object.

3. Create a header file named `"Color.h"` and define the class *Color* within the namespace *PS7*. The class must contain

   - ☐ a private int field named *red*.
   - ☐ a private int field named *green*.
   - ☐ a private int field named *blue*.
   - ☐ a public default constructor that assigns 0 to all fields.
   - ☐ a public copy constructor.
   - ☐ a public assignment operator.
   - ☐ a public empty destructor.
   - ☐ a public void setter method for *red* that assigns the parameter to *red* only if it is between 0 and 255 inclusively.
   - ☐ a public void setter method for *green* that assigns the parameter to *green* only if it is between 0 and 255 inclusively.
   - ☐ a public void setter method for *blue* that assigns the parameter to *blue* only if it is between 0 and 255 inclusively.
   - ☐ a public constant getter method for *red*.
   - ☐ a public constant getter method for *green*.
   - ☐ a public constant getter method for *blue*.
   - ☐ a public string constant method named `ToString()` that takes no parameters. It returns a string in the format

     $$\texttt{<}x\texttt{,}y\texttt{,}z\texttt{>}$$

     where x, y and z are the values of *red*, *green*, and *blue* respectively such that each value has three digits.
   - ☐ an ostream operator that returns an outcome in the same format as `ToString()`.

   Afterward, within the accompanying cpp file, within the main function, declare a *Color* object. Try to assign a couple of the fields of the object values outside the range 0 through 255, and then, display the object. Next, assign the fields of the object values in the range 0 through 255, and then, display the object.