Problem Set 09 - Polymorphism

Complete each task below. Remember to include all header files in the accompanying cpp file. Include the libraries *cctype* and *iomanip*.

Tasks:

1. Create a header file named `"Increaser.h"` and define the interface *Increaser* within the namespace *PS9*. The interface must contain

   - ☐ a void pure virtual method named `Increment()` that takes no parameters.
   - ☐ a bool pure virtual constant method named `CanIncrement()` that takes no parameters.

2. Create a header file named `"Decreaser.h"` and define the interface *Decreaser* within the namespace *PS9*. The interface must contain

   - ☐ a void pure virtual method named `Decrement()` that takes no parameters.
   - ☐ a bool pure virtual constant method named `CanDecrement()` that takes no parameters.

3. Create a header file named `"Alphabet.h"` and define the class *Alphabet* within the namespace *PS9*. The class must publicly inherit *Increaser* and *Decreaser* and contain

   - ☐ a private char field named *value*.
   - ☐ a public default constructor that assigns `'A'` to *value*.
   - ☐ a public copy constructor.
   - ☐ a public assignment operator.
   - ☐ a public empty destructor.
   - ☐ a public constant getter method for *value* named `GetValue()`.
   - ☐ a public setter method for *value* named `SetValue()` take assigns the parameter to *value* only if the parameter is an uppercase letter.
   - ☐ a public overridden `Increment()` method that makes *value* the next consecutive letter only if it is not `'Z'`.
   - ☐ a public overridden `CanIncrement()` method that returns true only if *value* is not equal to `'Z'`.
   - ☐ a public overridden `Decrement()` method that makes *value* the previous consecutive letter only if it is not `'A'`.
   - ☐ a public overridden `CanDecrement()` method that returns true only if *value* is not equal to `'A'`.
   - ☐ a public string constant method named `ToString()` that takes no parameters. It returns a string in the format

     `((`x`))`

     where x is the value of *value*.
   - ☐ an ostream operator that returns its outcome in the same format as `ToString()`.

4. Create a header file named `"HourTimer.h"` and define the class *HourTimer* within the namespace *PS9*. The class must publicly inherit *Increaser* and *Decreaser* and contain

   - ☐ a private int field named *value*.
   - ☐ a public default constructor that assigns o to *value*.
   - ☐ a public copy constructor.
   - ☐ a public assignment operator.
   - ☐ a public empty destructor.
   - ☐ a public constant getter method for *value* named `GetValue()`.
   - ☐ a public setter method for *value* named `SetValue()` take assigns the parameter to *value* only if the parameter is non-negative.
   - ☐ a public overridden `Increment()` method that increments *value* by 1 only if it is less than 3600.
   - ☐ a public overridden `CanIncrement()` method that returns true only if *value* is not equal 3599.
   - ☐ a public overridden `Decrement()` method that decrements *value* by 1 only if it is not equal 0.
   - ☐ a public overridden `CanDecrement()` method that returns true only if *value* is not equal to 0.
   - ☐ a public string constant method named `ToString()` that takes no parameters. It returns a string in the format

     `[`x`:`y`]`

     where x is `value / 60` with a preceding `'0'` if the quotient is less than 10 and y is `value% 60` with a preceding `'0'` if the remainder is less than 10.
   - ☐ an ostream operator that returns its outcome in the same format as `ToString()`.

5. In the accompanying cpp file, within the main function, declare a *Alphabet* object and a *HourTimer* object. And then, define

   - a bool function named `Increment()` that takes a *Increaser* reference parameter. If the parameter can be incremented, the function will increment the parameter and return true; otherwise, it returns false.
   - a bool function named `Decrement()` that takes a *Decreaser* reference parameter. If the parameter can be incremented, the function will increment the parameter and return true; otherwise, it returns false.