Lab 06 - Chain of Responsibility Design

Instructions:

- **The chain of responsibility design is a behavioral design pattern that lets you pass requests along a chain of handlers such that each handler decides to process the requests and/or pass them to the next handler in the chain.**
- **Your submissions must be submitted to the GitHub repository in the Lab06 directory.**
- **Within the directory is an accompanying header file named `"Helper.h"` that contains three constant strings named `LOWERS`, `UPPERS`, `DIGITS` that are equal to a string of all lowercase letters, a string of all uppercase letters, and a string of all digits respectively. And a bool function named `Contains()` that takes a string parameter and a char parameter respectively, and returns true if the char parameter is in the string parameter; otherwise, it returns false.**
- **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		1.0	
2		3.0	
3		1.0	
Total		5.0	

Task 1

Create a header file named `"Validator.h"` that defines an abstract class named *Validator* within the namespace *LB6* that must contain

- a private *Validator* pointer field named *next*.
- a public default constructor that assigns null to *next*.
- a public constant getter method for *next* named `GetNext()`.
- a public setter method for *next* named `SetNext()` that assigns the paramater to *next*.
- a public bool pure virtual constant method named `Validate()` that takes a string parameter.

Task 2

Create header file named `"Checks.h"` that define the class *CheckLength* that publicly inherit *Validator* within the namespace *LB6* and must contain

- a private unsigned int field named *length*.
- a private bool field named *bound*.
- a public default constructor that assigns 8 and `true` to *length* and *bound* respectively.
- a public overloaded constructor that takes an unsigned int parameter and a bool parameter respectively, and assigns the parameters to their respective fields.
- a public overridden `Validate()` method. If *bound* is `true` and the length of the parameter is at least the value of *length* or *bound* is `false` and the length of the parameter is at most the value of *length*, it returns `true` if *next* is equal to null; otherwise, it returns the return of the `Validate()` method of *next* with the parameter as its argument. Otherwise, it returns false.

and define the class *CheckCount* that publicly inherit *Validator* within the namespace *LB6* and must contain

- a private string field named *values*.
- a private unsigned int field named *count*.
- a public default constructor that assigns a string of all lowercase letters and 1 to *values* and *count* respectively.
- a public overloaded constructor that takes a string parameter and an unsigned int parameter respectively, and assigns the parameters to their respective fields.
- a public overridden `Validate()` method. If at least *count* amount of element of the parameter are in *values*, it returns true if *next* is equal to null; otherwise, it returns the return of the `Validate()` method of *next* with the parameter as its argument. Otherwise, it returns false.

  Hint: use `Contains()` from `"Helper.h"`.

and define the class *CheckCharacters* that publicly inherit *Validator* within the namespace *LB6* and must contain

- a private string field named *valids*.
- a public default constructor that assigns a string of all lowercase letters, uppercase letters, and digits to *valids*.
- a public overloaded constructor that takes a string parameter, and assigns the parameter to *valids*.
- a public overridden `Validate()` method. If the parameter contains only characters from *valids*, it returns true if *next* is equal to null; otherwise, it returns the return of the `Validate()` method of *next* with the parameter as its argument. Otherwise, it returns false.

  Hint: use `Contains()` from `"Helper.h"`.

Task 3

Create header file named `"Testing.h"` that defines a function named `ValidPass()` that takes a string parameter within the namespace *LB6*. It should return true if the length of the parameter is between 8 and 24 inclusively, with at least 2 uppercase letters, at least 2 lowercase letters, and at least 3 digits; otherwise, it should return false. It should do so without using loops and selection statements (if statements).

Extra Credit

In a header file named `"Extra.h"` define a class named *Badge* that must contain

- a private string field named *value*.
- a public default constructor that assigns the empty string to *value*.
- a public copy constructor.
- a public assignment operator.
- a public empty destructor.
- a friend overloaded equal operator that takes two constant *Badge* reference parameters and returns true only if their *value* fields are equal; otherwise, returns false.
- a public constant getter method for *value* named `GetNumber()`.
- a public setter method for *value* named `SetNumber()` that assigns the parameter to *value* only if the length of the parameter is exactly 8 digit characters with at least half of them odd digits.
- a public string constant method named `ToString()` that takes no paramaters. It returns a string of 8 asterisks if *value* is an empty string; otherwise, it returns *value*.
- a friend overloaded ostream operator that returns an output in the same format as `ToString()`.

(2 points)