Instructions

* Download and extract the .zip file with program source code and directions.
* Complete the *Test* functions in the *Submission.cpp* file.
* While working, Debug/Run the lab to see how many test runs pass or fail.
* Ideally, you want all test runs to pass to get the maximum amount of points.
* Your grade is shown at the bottom when you run the program.
* **You cannot use auto in place of known data types in PG1.**

**You cannot use multiple return statements in any method submitted for this project.** Additionally, the use of auto is not permitted. You must write the necessary programming statements/logic to accomplish each task. No credit will be given for using existing methods (thereby avoiding the need to write the programming statements) to accomplish the assigned tasks.

Circle Class

You will write your own class, Circle, make an instance of it, and return it. That is, you will return an object that you make of type Circle. In other words, if you were to make an object of type Circle called c, you would return c from this Test. You should create the Circle class within its own file named Shape.cpp.  
  
Your Circle class must be imported within your Submission.cpp file, so once it is completed, uncomment the corresponding lines.

To create your Circle class, add the following access modifier, fields, constructor, and methods. In addition, add the line #define SHAPE at the top of Shape.cpp

For example:

#define SHAPE

class Circle

{

// fields of this class

// constructors of this class

// methods of this class

}

Your Circle class must have three member variables (fields) of type int, which represent:

* the x position of the Circle
* the y position of the Circle
* the radius of the Circle

You can name the fields using any name you want, though you could use the following:

* int mX;
* int mY;
* int mRadius;

The prefix 'm' stands for a member variable.  
  
Your Circle class must have an overloaded constructor, which takes three variables that will be assigned to the fields of the class:

Circle(int x, int y, int radius)

Remember: a constructor *must be* under a public access heading.

This overloaded constructor should take the variables passed into the class and assign them to the fields.

Do not include a default constructor.  
  
Your Circle class must have the following three accessor (getter) methods, each of which returns one of the three fields of the class. Each accessor method must return the appropriate field.

int GetX()

int GetY()

int GetRadius()

For Tests 1 through 4, when you have finished adding the above-requested information to your Circle class, you will then make an instance object of type Circle, calling its overloaded constructor and passing the variables given to you for each Test, and then you will return your class instance object from this Test.

For example:

Circle c1(x, y, radius);

return c1;

TextCodec Class

For tests 5 and 6, you will end up writing your own class, called TextCodec, make an instance of it, call the Encode() method (described below), and return the result of that method. The TextCodec class should be included within a new TextCodec.cpp file.

To create your TextCodec class, you will add the following access modifier, fields, constructor, and methods.

In addition, add the line #define TEXT\_CODEC at the top of TextCodec.cpp

For example:

#define TEXT\_CODEC

class TextCodec

{

// fields of this class

// constructors of this class

// methods of this class

}

Your TextCodec class must have one member variable (field), which is:

* a short int, which holds a numeric offset

You can name the field any name you want.

Your TextCodec class must have an overloaded constructor, which takes the offset variable that will be assigned to the field of the class:

TextCodec(short int offset)

Encode Method

Your TextCodec class must have the following public method:

string Encode(string message)

The purpose of the Encode() method is to take the message passed into the method and encode it into an unreadable string using the offset field stored in the class. To accomplish this, we will provide you with the following code to use as the body of your Encode method:

std::string result = message;

for (int i = 0; i < message.length(); i++)

{

result[i] = (char)(result[i] + offset);

}

return result;

For this Test, when you have finished adding the above-requested information to your TextCodec class, you will then make an instance object of type TextCodec, calling its overloaded constructor and passing the offset variable given to you in this Test, and then you will call the Encode() method passing the message variable given to you in this Test, and return the result of the Encode() method.

For example:

TextCodec codec(offset);

// call the Encode() method passing message and return the result

Decode Method

Add the following public method to your TextCodec class:

string Decode(string message)

The purpose of the Encode() method is to take the message passed into the method and decode it back to a readable string using the offset field stored in the class. The decoding code works nearly exactly the same as the encoding code, except you will subtract the offset from each character in the string instead of adding it.

For this Test, when you have finished adding the above-requested information to your TextCodec class, you will then make an instance object of type TextCodec, calling its overloaded constructor and passing the offset variable given to you in this Test, and then you will call the Decode() method passing the message variable given to you in this Test, and return the result of the Decode() method.

For example:

TextCodec codec(offset);

// call the Decode() method passing message and return the result

Book Class

For tests 7 and 8 you will end up writing your own class, called Book, make an instance of it, and return the result of that method. The Book class should be included in a new Book.cpp file.

To create your TextCodec class, add the following access modifier, fields, constructor and methods.

In addition, add the line #define BOOK at the top of Book.cpp

For example:

#define BOOK

class Book

{

// fields of this class

// constructors of this class

// methods of this class

}

Your Book class must have two member variable (field), which is:

* a string, which holds a title
* a string, which holds an author

You can name the field any name you want.

Your book will have a default constructor and an overloaded control. By default, the default constructor should:

* Set the title to "Moby-Dick"
* Set the author to "Herman Melville"

In addition, your book should have an overloaded constructor that accepts both an author and a title and sets the appropriate fields.

Finally, your book will need accessors for both the Title and Author. Remember, these must both be under the public heading to be accessible.

For example:

public:

string GetAuthor(){

// ...

}

string GetTitle(){

// ...

}

For these Test, when you have finished adding the above-requested information to your Book class, you will then make an instance object of type Book, calling the appropriate constructors.

For example:

Book book1;

return book1;

Book book2(title, author);

return book2;

Rubric

1. Test 1 - Return the exact requested result - 12.5%
2. Test 2 - Return the exact requested result - 12.5%
3. Test 3 - Return the exact requested result - 12.5%
4. Test 4 - Return the exact requested result - 12.5%
5. Test 5 - Return the exact requested result - 12.5%
6. Test 6 - Return the exact requested result - 12.5%
7. Test 7 - Return the exact requested result - 12.5%
8. Test 8 - Return the exact requested result - 12.5%

Deductions

* **One-time 10pt deduction:** Any method contains multiple return statements.
* **One-time 10pt deduction:**Submission does not contain detailed comments, using appropriate terminology to explain code solutions.

Deliverables

When you are ready to submit:

* Close *Visual Studio*
* **Do not change any file names**
* Compress (".zip up") your "Lab#" source code folder into a .zip file
* Upload a zip file named ***LastFirst\_Lab7.zip*** containing all necessary files.
  + MartinMarty\_Lab7.zip
* Make sure your program compiles and runs - submitting a program that does not compile and run will get you 0 points