Program Numerical Palindromes

This project will examine the concept of a numerical palindrome implemented using classes. A numerical palindrome is a non-negative whole number that is the same forwards and backwards, e.g. 2332, 12321, etc.

Start with any positive whole number. If it's a palindrome, you are done, if it isn't, reverse the number, add the reversal to the original value, and test this new result. If it is a palindrome, you are done; otherwise repeat the process: reverse, add, and test again, and so forth. The process almost always leads, very quickly, to a palindrome.

Example:
- 152 (no)
- 152 + 251 = 403 (no)
- 403 + 304 = 707 (yes)

Another Example:
- 552 (no)
- 552 + 255 = 807 (no)
- 807 + 708 = 1515 (no)
- 1515 + 5151 = 6666 (yes)

For this assignment implement, NumPal.cpp using the NumPal.hpp specification file that would compile and run with the main.cpp. The implementation file will allow the main.cpp to produce the following output after accepting an initial (string) the input value.

Here is the sample output for 552

```
Start value is 552
552 reverse-> 255
new sum: 552 + 255 = 807
807 reverse-> 708
New sum: 807 + 708 = 1515
1515 reverse-> 5151
new sum: 1515 + 5151 = 6666
final value: 6666

6666 is a palindrome
number of pass is 3.
```

As indicated above, this process does not always work. For example, 196 goes on and on. Thus, it is necessary to cap the number of iterations. For this project we cap the number of pass to 10.

In addition to implementing NumPal.cpp comment out the error handling in main.cpp and write code to throw exception for a negative start value and start value that exceed 5 characters. The following is a list of test cases:

1. 121,
2. 1221
3. 45754
4. 152
5. 552 (more test case value continued on the next page)

6. 997
7. 196
8. -123
9. 123456

The other definition of a numeric palindrome is all numbers that have the same digits such as 4, 11, 55, 222, and 6666 are examples of palindromes. Therefore, create a special class to test out numbers by inheriting from the base class, NumPal.

Here is the list of deliverables for this project.

1. Implementation of NumPal.cpp
2. Implement Friend.cpp
3. Modification to main.cpp for exceptional handling of input errors,
4. Inheritance class for testing on special palindrome.

Program Caesar Cipher

Write a C++ program that consists of a driver (main.cpp), CaesarCipher.cpp (class implementation) and CaesarCipher.hpp (class specification/declaration).

For the Caesar Cipher class implementation, you can use any data structure that we covered in class. The Caesar Cipher class must have a Friend method that can be used by another cipher class that will have access the Caesar cipher set of characters. The set of characters can be a static member variable.

The class implementation must have the following:

- Cipher class constructor,
- User-defined class constructor for the number to shift for the encrypt text.
- Destructor to free up the data structure if needed.
- The ability to read and write to a text file. If a text file does NOT exist create one.
- The ability to read and write to a binary file. If a binary file does NOT exist create one.
- Error handling can be implemented by throwing an exception or display an error message to the console.
- Copy constructor that copy your shifted data structure.
- A method for encrypting an encrypted message.
- A method for decrypting the message.

The main.cpp will ask the user how to encrypt the message either from a file or from the console.

The data set of characters are A-Z, a-z, 0-9.
Any other special characters within the message like "[", "]", etc. will not get encrypted or decrypted.