CSCI 335
First programming assignment (100 points)
Due September 6
Please follow the blackboard instructions on writing and submitting programming assignments. It should compile to receive any credit.
Make sure to include a README.txt file with your submission where you state what you have completed.

Programming: The big five

Create and test a class called **Chain**. A chain is just a series of items, e.g. [2 7 -1 43] is a chain containing four integers. A Chain can have any size. An empty Chain has size 0.

The purpose of this assignment is to have you create a Chain class from scratch without using the STL. Since Chain can have arbitrary size, you should use pointers. The private data members should be:

size_t size_; Object *array_;

Object is the template type parameter.

Pay special attention to Weiss's **"big five",** the destructor, copy constructor, copy assignment operator, move constructor and move assignment operator.

Included are all the two files (chain.h and test_chain.cpp) you will need, as well as the Makefile. Do not modify the Makefile or the file names. Do not modify the test_chain.cpp file except by changing or adding include files if needed. You can comment in the main file the parts you didn't complete. The chain.h file is not complete. In the file it is explained where to provide changes. I am also providing you with a sample input file `test_input_file.txt` and I am explaining how to use it at the end of this document.

This assignment will help you revisit constructors, destructors, overloading of operators, and templates. Follow a consistent C++ coding style, for instance
https://google.github.io/styleguide/cppguide.html

PART 1 [60 points]

Implement the "big five". Add the output stream << operator.

Demonstrate that you are able to read and write correctly by including the following code in the main file. The code is already provided for you in the main file. You can comment parts of it as you are testing your implementation. For full credit all functions should work.

```
void TestPart1() {

  Chain<int> a, b;  // Two empty Chains are created
  cout << a.Size() << " " << b.Size() << endl; // yields 0 0
```

```
  Chain<int> d{7};  // A chain containing 7 should be created.

  cout << d;  // Should just print [7]

  a.ReadChain();
 // User enters a chain, for example [4: 10 30 -1 2]
 // 4 provides the size of the chain, after the : four items
 // are included. Note that you should also expect [ and ].
 // You can abort() the routine if the input is not of the required format.

  cout << a;  // Output should be what user entered.

  b.ReadChain();  // Same for b.

  cout << b;

  Chain<int> c{a};  // Calls copy constructor for c.
  cout << c;
  cout << a;

  a = b;  // Should call the copy assignment operator for a.
  cout << a;

  Chain<int> e = std::move(c);  // Move constructor for d.
  cout << e;
  cout << c;

  a = std::move(e);  // Move assignment operator for a.
  cout << a;
  cout << e;
}
```

PART 2 [40 points]

Overload the + and [] operators for your Chain class. Test with the following code. The code is already provided for you in the main file. You can comment parts of it as you are testing your implementation. For full credit all functions should work.

```
void TestPart2() {

  Chain<string> a, b;

  a.ReadChain();  // User provides input for Chain a.

  cout << a;

  b.ReadChain();  // User provides input for Chain b.
  cout << b << endl;

  cout << a + b << endl; // Concatenates the two Chains.


  Chain<string> d = a + b;
```

```
  cout << d;

  cout << d + "hi_there";  // Adds an element to the end.


  cout << a[2] << endl;  // Should print the 3rd element.
                         // Throw an exception (or even abort()) if is out of
                         // range.
  b[1] = "b_string";  // Should change the 2nd element to "b_string"
  cout << b;
  b[0] = "a_string";
  cout << b;


}  // End of TestPart2
```

Do not send to the standard output anything else other what is asked. Do not pause for input, or interact with the user.

I will run your program as follows:

```
test_chain < test_input_file.txt
```

In that case the output should be the one contained in `expected_output.txt`
I may also test it with other input configurations.