CSCI 335 – Spring 2021

Homework #2
First C++ programming assignment (100 points)
Due February 23, 11:00pm
This is an individual assignment

Please follow the Blackboard instructions on writing and submitting programming assignments. The functions you provide should compile and run to receive any credit.

Make sure to include a README.2D file with your submission where you state what you have completed.

Programming: The "big five"

Create and test a class called **Points2D**. This class describes a sequence of 2D points (i.e. points in the 2D-plane). For example (1, 3), (4, 5) is a sequence of two points, where each coordinate is an integer. (1.2, 3.4), (5.6, 10.1), (11.1, 12.0) is a sequence of three points where each coordinate is a double. A sequence can have any size. An empty sequence has size 0.

The purpose of this assignment is to have you create a Points2D class from scratch with limited help from the STL. Since Points2D can have arbitrary size, you should use pointers. The private data members should be:

```
size_t size;  std::array<Object, 2> *sequence_;
```

Object is the template type parameter (i.e. int, double, etc.). An initial piece of code with the structure of the class is provided. Do not change the data representation (for instance do not use a vector or list to represent the sequence_).

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

Included are the two files (points2d.h, test_points2d.cc) you will need, as well as the Makefile. Do not modify the Makefile or the file names. Do not modify the test_points2d.cc file except by changing or adding include files if needed. You can comment in the main file the parts you didn't complete. The points2d.h file is not complete. In the file it is explained where to provide changes. You will be provided with a sample input file `test_input_file.txt` and it is explained at the end of this document how to use it.

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 [70 points]

Implement the “big five”. Add the output stream << operator.

Demonstrate that you are able to read and write data 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() {
  Points2D<int> a, b;  // Two empty Points2 are created.
  cout << a.size() << " " << b.size() << endl; // yields 0 0.
  const array<int, 2> a_point2d{{7, 10}};
  Points2D<int> d{a_point2d}; // Sequence (7, 10) should be created.
  cout << d;  // Should just print (7, 10)
  cout << "Enter a sequence of points (integer)" << endl;
  a.ReadPoints2D(); // User enters a set of points in the form:
                // 3 7 4 3 2 1 10
                // 3 specifies number of points. Points are the pairs
                // (7, 4) (3, 2) and (1, 10)
  cout << "Output1: " << endl;
  cout << a;  // Output should be what user entered.
  cout << "Enter a sequence of points (integer)" << endl;
  b.ReadPoints2D();  // Enter another sequence.
  cout << "Output2: " << endl;
  cout << b;
  Points2D<int> c{a};  // Calls copy constructor for c.
  cout << "After copy constructor1 c{a}: " << endl;
  cout << c;
  cout << a;
  a = b;  // Should call the copy assignment operator for a.
  cout << "After assignment a = b" << endl;
  cout << a;
  Points2<int> e = move(c);  // Move constructor for d.
  cout << "After e = move(c) " << endl;
  cout << e;
  cout << c;
  cout << "After a = move(e) " << endl;
  a = move(e);  // Move assignment operator for a.
  cout << a;
  cout << e;
}
```

PART 2 [30 points]

Overload the + and [] operators for your Points2D 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() {
Points2D<double> a, b;
  cout << "Enter a sequence of points (double)" << endl;
  a.ReadPoints2D();  // User provides input for Points2D a.
  cout << a;
  cout << "Enter a sequence of points (double)" << endl;
  b.ReadPoints2D();  // User provides input for Points2 b.
  cout << b << endl;
  cout << "Result of a + b" << endl;
  cout << a + b << endl;
  Points2D<double> d = a + b;
  cout << "Result of d = a + b" << endl;
  cout << d;
  cout << "Second element in a: " << endl;
  cout << a[1][0] << ", " << a[1][1] << endl;  // Should print the 2nd element.
}  // 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.

You can run the program as follows:

./test_points2d

In that case you should enter the required data in the standard input.

You can also run it as follows

```
./test_points2d < test_input_file.txt
```

In that case the output should be the one contained in `expected_output.txt`

Note that the code for `test_points2d` is the same in both cases. In the second call the shell redirects the contents of `test_input_file.txt` to the standard input.

Deliverables to upload to Gradescope for Homework 2:

- Submit your modified points2d.h with inline comments.
- A single README.2D text file where you state in your own words what you have completed for parts 1 and 2. Please make sure to add your name at the top of this file.