CSCI 335 **Spring 2020**

HW 1: C++

100 points

Due 11pm February 13, 2020

This is an individual assignment

Instructions:

1. **Read and follow the contents of 335 Spring20_335_Assignments_Instructions document on the blackboard.**
2. **Read the assignment description below.**
3. **Submit only the files requested in the deliverables at the bottom of this description to gradescope by the deadline.**

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

Question 1: 15 points

Write the routines with the following declarations:

```
void permute( const string &  str );
void permute( const string & str, int low, int high );
```

The first routine is a driver that calls the second and prints all the permutations of the characters in string str. If str is "abc", then the strings that are output are abc, acb, bac, bca, cab and cba. Use recursion of the second routine.

Question 2: 85 points

For this question, you must create and test a class called **Points2**. This class describes a sequence of 2D points.
Example 1: (1, 3), (4, 5) is a sequence of two points, where each coordinate is an integer.
Example 2: (1.2, 3.4), (5.6, 10.1), (11.1, 12.0) is a sequence of three points where each coordinate is a double. The sequence can be of arbitrary size and can . An empty sequence has size 0.

The purpose of this assignment is to have you create a Points2 class from scratch with limited help from the STL. Since Points2 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 Weiss's **"Big-Five",** the destructor, copy constructor, copy assignment operator, move constructor and move assignment operator.

Included are the two files (points2.h, test_points2.cc) you will need, as well as the Makefile. Do not modify the Makefile or the file names. Do not modify the test_points2.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 points2.h file is not complete. The file provides details on where to provide changes must be made. You are also provided with a sample input file test_input_file.txt and an explanation on how to use it is provided at the end of this document.

PART 1 [55 points]

Implement the "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 test your implementation. For full credit all functions should work.

```
void TestPart1() {

  Points2<int> a, b;  // Two empty Points2 are created.

  cout << a.size() << " " << b.size() << endl; // yields 0 0.
  const array<int, 2> a_point2{{7, 10}};
  Points2<int> d{a_point2}; // Sequence (7, 10) should be  created.
  cout << d;  // Should just print (7, 10)
  cout << "Enter a sequence of points (integer)" << endl;
  a.ReadPoints2(); // 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.ReadPoints2();  // Enter another sequence.
  cout << "Output2: " << endl;
  cout << b;
  Points2<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 Points2 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() {
Points2<double> a, b;
   cout << "Enter a sequence of points (double)" << endl;
   a.ReadPoints2();  // User provides input for Points2 a.
   cout << a;
   cout << "Enter a sequence of points (double)" << endl;
   b.ReadPoints2();  // User provides input for Points2 b.
   cout << b << endl;
   cout << "Result of a + b" << endl;
   cout << a + b << endl;
   Points2<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_points2**
In that case you should enter the required data in the standard input.
You can also run it as follows: **./test_points2 < test_input_file.txt**
In that case the output should be the one contained in expected_output.txt
Note that the code for test_points2 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 Assignment 1:

- Question 1: Submit your modified **permute.h**[1] with inline comments.
- Question 2: Submit your modified points2.h with inline comments.

[1] *Submit the routines as a header file permute.h. It **should not** contain a main file. This is required due to gradescope constraints.

- A single README file where you state what you have completed for each question.