**Project Plan**

**Purpose**

Design and construct a program to iterate through all permutations of a set of integers.

**Deliverables**

**PermutationGenerator.h** header file, defining a class with the following methods:

* PermutationGenerator(int n) -- Constructor which initializes the internal data to the first permutation (1, 2, 3, …, n). Throws exception of n < 0 or n > 25.
* bool next() -- Transformer method which performs a swap in the internal data, arriving at the next permutation. If successful, returns true. Returns false if there are no more permutations to be had.
* void printCurrent() -- Observer method which prints the current permutation to standard output, with spaces between the numbers, e.g  
  1 3 2
* Int getElement(int i) -- Observer method which returns the integer at position ‘i’ of the current permutation. Throws an exception if i < 0 or i >= n.

**PermutationGenerator.cpp** implementation file, containing the implementation.

**AutoTester.cpp** program which runs through a set sequence of tests, and outputs any failures. Verifies the following:

* Constructor works for n = 3, and throws exceptions where appropriate.
* getElement(i) throws exceptions when ‘i’ is out of bounds.
* next() correctly iterates through all 6 permutations, when n = 3, in the following order:

1 2 3

1 3 2

3 1 2

3 2 1

2 3 1

2 1 3

* next() returns false when n = 3, after the 6th permutation.

**ListPermutations.cpp** command-line program which takes an integer n from the command line, and prints out n followed by all the permutations of (1, …, n) in order, e.g.

./ListPermutations 4

4

1 2 3 4

1 2 4 3

.

.

.

**CheckUnique.cpp** command-line program which reads lines from standard input, in the format shown above. After reading the value of ‘n’, it reads every subsequent permutation line and counts them. The program verifies the following:

* Each line contains only the integers {1, …., n}
* Each line is unique
* There are n! Total permutation lines.

**Team Members/ Roles**

Thomas Swinburne – QA

Aaron Garten – Tech Lead

Terry Ford- Project Manager

**Timeline**

Initial meeting on project 8/29/2019

Coding of the Project 8/30-

**Testing Plan**

The AutoTester program can be run quickly and often, and should always run with no errors. The ListPermutations program can be run on bigger values of n for manual inspection and debugging purposes. For large values of n, the CheckUnique program can be used as a final verification step. We do not expect to test the program on every allowable value of n, but values of n under 10 should be fully verifiable.