**Programming Assignment #4**

Tara Moses

November 20, 2015

**Design:**

This program was designed to sort 100 random integers in the range [1, 10 000] using two different sorting methods. The program doesn't take in input. 100 numbers are randomly generated, stored in an array, and sorted using Corn Sort first and Hi-Low Sort next. The unsorted data and both sets of sorted data are saved in text files labeled “unsorted.txt”, “cornsorted.txt”, and “hilowsorted.txt”, respectively.

**Implementation:**

The program was implemented using an array of 15 random numbers in the range [1, 20] for testing. Also, instead of saving the arrays to text files, the arrays were printed out to the screen at first. Once both sorting functions were perfected, the array sizes and max values were changed, and the arrays were copied to text files. There were two unsorted arrays: one for Corn to sort, and one for Hi-Low to sort. Each of these unsorted arrays contained the same random numbers.

Timing for Corn Sort was O((2n-2)^n) because each for-loop pass runs through n-1 numbers, and you have to do, at worst-case, n passes (worst-case involves the biggest number being at the very left, for example). Timing analysis for Hi-Low sort was O(2n \* (2n-2) \* (2n-4) \* (2n-6) \* … \* 2 \* 1) due to each for-loop (there are two per pass) running through the range (high – low). (High – low) starts at (SIZE-1 – 0) = n integers and goes down until (high – low) = 1.

**Testing:**

The assignment compiled and ran on a computer using Netbeans 8.0.1 on Linux Mint XFCE. There were no inputs. Output text files were used for each sorting function and the unsorted set of integers.

**Summary:**

Overall, the assignment was a success. The program successfully sorts the integers and saves them to text files. For the test cases, in order to show that the sorting functions worked, the arrays were shortened to 15 numbers and the numbers were between 1 and 20 inclusive. However, the finished program involved arrays of 100 integers between [1, 10 000].

**Test Case 1:**

Unsorted:

3 20 12 2 12 16 16 4 19 20 10 9 14 17 3

Corn Sort:

2 3 3 4 9 10 12 12 14 16 16 17 19 20 20

Hi-Low Sort:

2 3 3 4 9 10 12 12 14 16 16 17 19 20 20

------------------

(program exited with code: 0)

Press return to continue

**Test Case 2:**

Unsorted:

7 2 6 13 5 19 8 17 3 4 5 8 10 10 4

Corn Sort:

2 3 4 4 5 5 6 7 8 8 10 10 13 17 19

Hi-Low Sort:

2 3 4 4 5 5 6 7 8 8 10 10 13 17 19

------------------

(program exited with code: 0)

Press return to continue

**Test Case 3:**

Unsorted:

13 15 1 11 13 2 6 10 4 13 14 16 9 4 15

Corn Sort:

1 2 4 4 6 9 10 11 13 13 13 14 15 15 16

Hi-Low Sort:

1 2 4 4 6 9 10 11 13 13 13 14 15 15 16

------------------

(program exited with code: 0)

Press return to continue