Equivalence Class Testing & Decision Table Testing of Sort.exe

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Θ(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Θ(n log(n))	Θ(n log(n))	Θ(n log(n))	O(n)
Timsort	Θ(n)	Θ(n log(n))	Θ(n log(n))	O(n)
Heapsort	Θ(n log(n))	Θ(n log(n))	Θ(n log(n))	O(1)
Bubble Sort	Θ(n)	Θ(n^2)	Θ(n^2)	O(1)
Insertion Sort	Θ(n)	Θ(n^2)	Θ(n^2)	O(1)
Selection Sort	Θ(n^2)	Θ(n^2)	Θ(n^2)	O(1)
Tree Sort	Θ(n log(n))	Θ(n log(n))	Θ(n^2)	O(n)
Shell Sort	Θ(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Θ(n+k)	Θ(n+k)	Θ(n^2)	O(n)
Radix Sort	Θ(nk)	Θ(nk)	Θ(nk)	O(n+k)
Counting Sort	Θ(n+k)	Θ(n+k)	Θ(n+k)	O(k)
Cubesort	Θ(n)	Θ(n log(n))	Θ(n log(n))	O(n)

Table of Contents

Project Overview

In this project we test Sort.exe using Traditional, Normal, Weak Robust, and Strong Robust Equivalence class testing, as well as using Decision Table Testing. These testing methods are examples of functional, or black box, testing, which means we do not have access to the code and must come up with a variety of test cases to test a multitude of different possible points of failure. All of our tests are run by hand (with a program to format the report) so it would be unfeasible to test every possible use case, to avoid testing every single possible use case we use the program specifications to determine the most likely points of failure. Depending on the type of testing strategy we pick valid values, invalid values, or a mix of both. The valid values we chose were negative integers, an integer just above the minimum integer, an integer just smaller than the maximum integer, and a list with several repeated integers. The invalid values we chose were doubles, fractions, characters, a list with no delimiters, a list with letters as the delimiters, a list with the numbers on separate lines, an integer just smaller than the minimum integer, and an integer just larger than the max integer. The combination of invalid and valid values allowed us to test a value from each side of every possible point of failure.

Test Plan

Equivalence Class Testing:

Traditional:

For traditional Equivalence Class Testing, the only values tested are invalid. These include: a list with no delimiter, greater than / less than 50 elements, a list using letters as delimiters, a list on different lines, smaller than minimum value / greater than the maximum value, doubles, and fractions.

Normal:

For normal Equivalence Class Testing, the only values tested are valid. These include: a list with a value just greater than the minimum value, a list with a value just smaller than the maximum value, a list with all negative numbers, a list with the same value multiple times

Weak Robust:

For Weak Robust Equivalence Class Testing, we test all valid values, with one invalid value. These include: One alphabet character in place of an integer, smaller than minimum value / greater than the maximum value, one double, and one fraction.

Strong Robust:

For Strong Robust Equivalence Class Testing, we test combinations of invalid values. These include: One alphabet character and one value smaller than the minimum, one alphabet character and one larger than the maximum, one alphabet character and one double, one alphabet character and one fraction, one smaller than the minimum and one larger than the maximum, one smaller than the minimum and one double, one smaller than the minimum and one fraction, one larger than the maximum and one double, one larger than the maximum and one fraction, and one double and one fraction.

Decision Table Testing:

For Decision Table-Based Testing we identify our conditions and then make a truth table for all test cases. The conditions we will be examining with these tests will be doubles, characters, negatives. In order to examine all combinations of the conditions we will have 8 test cases. These test cases will cover a wide variety of tests to expose any possible points of failures. When it comes to decision tables, there are 3 main parts with those being Limited Entry Decision Tables, Extended Entry Decision Tables, Mixed Entry Decision Tables

This Page is Intentionally Left Blank

Test Cases

Equivalence Class Testing

Insertion Sort:

Traditional:

Test Case #000:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with no delimiter

Preconditions: N/A

Inputs:

391410016932462685276864815418355183074731446781702753596119563588587298386494695046786578926207949

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
nsertion Sort: -2027713500,-2027713500,-2027713496,-2027713432,-2027713312,-202
713296,-456317600,-452933060,-452810152,-452783344,-452783344,-452611364,-45060
784,-450603336,-450599548,-450599536,-450555012,-450553879,-450360592,0,0,0,0,0
0,37,32645,32645,32645,32645,32645,32645,32645,32645,32645,32645,32645,32645,32
45,32766,32766,32766,32766,32766,32766,32766,3167716,12721243,814159578,2147483
47,

. Load File
. Exit Program
lease enter your selection: 2
```

Test Case #001:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with greater than 50 elements

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49,**101**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**101**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,

1. Load File
```

<u>Test Case #002:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with less than 50 elements

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,32766,

1. Load File
2. Exit Program
```

Test Case #003:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that uses letters as delimiters

Preconditions: N/A

Inputs:

39W14V100U16T93S24R62Q68P52O76n86M48L15K41J83I55H18G30F74E7D31C44B67A81Z70Y27

X53W59V61U19T56S35R88Q58P72O98N38M64L94K69J50I46H78G6F57E89D26C20B79A49

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69

,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -1133415072,-1130030532,-1129907624,-1129880816,-1129708836,-112
7700808,-1127697020,-1127697008,-1127696048,-1127652484,-1127651351,-1127458064,
-1127458064,-201850652,-201850652,-201850648,-201850584,-201850464,-201850448,0,
0,0,0,0,37,39,32727,32727,32727,32727,32727,32727,32727,32727,32727,32727,32727,
32727,32727,32727,32766,32766,32766,32766,32766,32766,32766,3167716,12721243,814
159578,
```

Test Case #004:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers from a text file containing data on different lines

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30

,74,7,31,44,67,81,70,27,53,59,61,19,

56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89

,26,20,79,49

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69

,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Passed(10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,19,24,26,27,30,30,31,35,38,39,41,44,46,48,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,89,93,94,98
,100,

1. Load File
```

<u>Test Case #005:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values greater than the maximum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**2147483648**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,2147483647,

1. Load File
```

Test Case #006:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values lower than the minimum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**-2147483649**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,
46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89
,93,94,98,100,

1. Load File
```

Test Case #007:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are doubles.

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**79.32,49.12**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,**49.12**,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,**79.32**,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,

1. Load File
```

<u>Test Case #008:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are fractions

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,**7/3**,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49

Expected Output:

7/3,6,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,

1. Load File
2. Exit Program
```

Normal:

Test Case #009:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are just greater than the minimum value

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,-**2147483648**

Expected Output:

-**2147483648**,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Passed(10/9/24)

```
Insertion Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,
46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89
,93,94,98,100,

1. Load File
2. Exit Program
Please enter your selection: 2
```

Test Case #010:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are just smaller than the maximum value

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**2147483647**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**2147483647**

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,2147483647,

1. Load File
```

<u>Test Case #011:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are all negative

Preconditions: N/A

Inputs:

-39,-14,-100,-16,-93,-24,-62,-68,-52,-76,-86,-48,-15,-41,-83,-55,-18,-30,-74,-7,-31,-44,-67,-81,-70,-27,-53,-59,-61,-19,-56,-35,-88,-58,-72,-98,-38,-64,-94,-69,-50,-46,-78,-6,-57,-89,-26,-20,-79,-49

Expected Output:

-100,-98,-94,-93,-89,-88,-86,-83,-81,-79,-78,-76,-74,-72,-70,-69,-68,-67,-64,-62,-61,-59,-58,-57,-56,-55,-53,-52,-50,-49,-48,-46,-44,-41,-39,-38,-35,-31,-30,-27,-26,-24,-20,-19,-18,-16,-15,-14,-7,-6,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Insertion Sort: -100,-98,-94,-93,-89,-88,-86,-83,-81,-79,-78,-76,-74,-72,-70,-69
-68,-67,-64,-62,-61,-59,-58,-57,-56,-55,-53,-52,-50,-49,-48,-46,-44,-41,-39,-38
-35,-31,-30,-27,-26,-24,-20,-19,-18,-16,-15,-14,-7,-6,

1. Load File
```

<u>Test Case #012:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that has repeated entries

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,**89,89,79,79,79**

Expected Output:

6,7,14,15,16,18,19,24,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,79,79,81,83,86,88,89,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Please enter your selection: 1


Insertion Sort: 6,7,14,15,16,18,19,24,27,30,31,35,38,39,41,44,46,48,50,52,53,55,
56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,79,79,81,83,86,88,89,89,93,94,98
,100,

1. Load File
2. Exit Program
```

Weak Robust:

Test Case #013:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that alphabetical

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**A**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,5
2,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,
100,
```

Test Case #014:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value greater than the maximum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**2147483648**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,2147483647,

1. Load File
```

Test Case #015:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value lower than the minimum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,-**2147483649**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,

64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,
46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89
,93,94,98,100,

1. Load File
```

Test Case #016:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that is a double.

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**49.12**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,**49.12**,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,
```

<u>Test Case #017:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that is a fraction

Preconditions: N/A

Inputs:

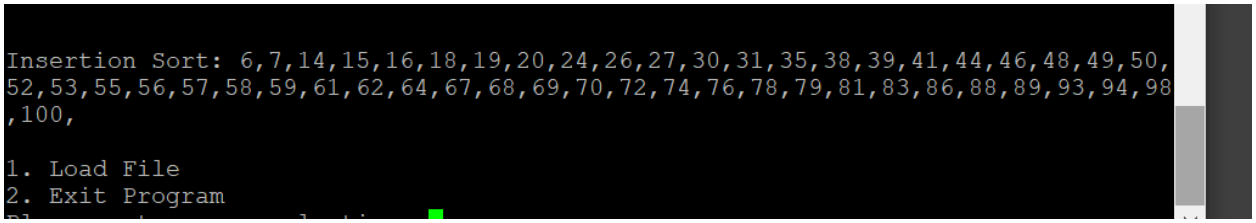
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,**7/3**,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49

Expected Output:

7/3,6,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



<u>Strong Robust:</u>

<u>Test Case #018:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one value smaller than the minimum.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**A**,**-2147483649**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

<u>Test Case #019:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one larger than the maximum.

Preconditions: N/A

Inputs:

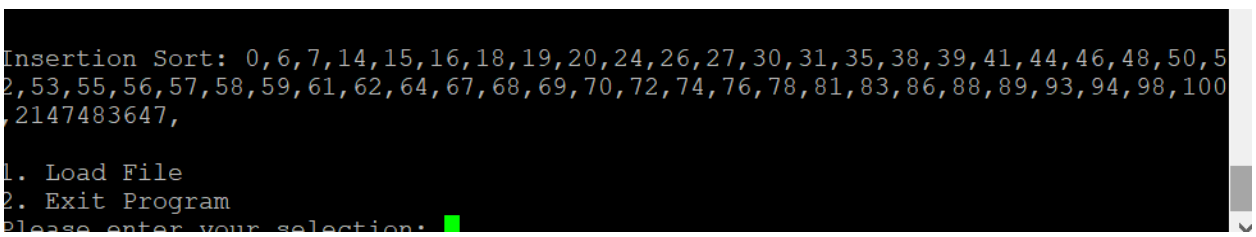
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20**,A,2147483648**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, **2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



Test Case #020:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one double.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**A,101.5**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, **101.5**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,5
2,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100
,101,

1. Load File
2. Exit Program
Please enter your selection:
```

Test Case #021:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20**,A,300/2**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,81,83,86,88,89,93,94,98,100, **300/2**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,5
2,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100
,300,

1. Load File
```

Test Case #022:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one larger than the maximum.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,**-2147483649,2147483648**

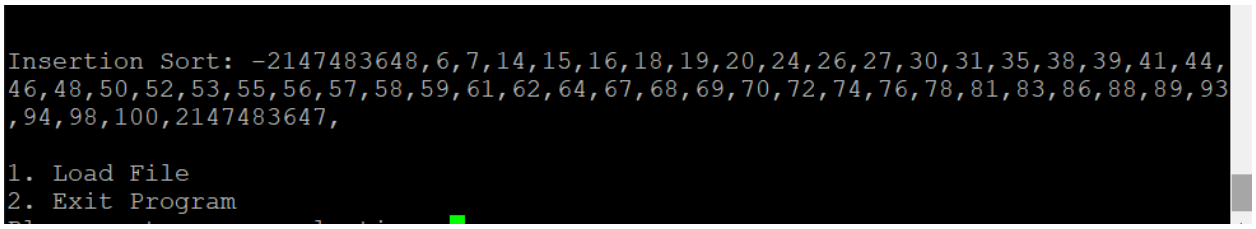
Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,

64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, **2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



<u>Test Case #023:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one double.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20**,-2147483649,101.5**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,

64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, **101**.5

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,
46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93
,94,98,100,101,

1. Load File
2. Exit Program
```

Test Case #024:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,**-2147483649, 300/2**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,

64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100**, 300/2**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,
46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93
,94,98,100,300,

1. Load File
2. Exit Program
Please enter your selection:
```

Test Case #025:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one larger than the maximum and one double.

Preconditions: N/A

Inputs:

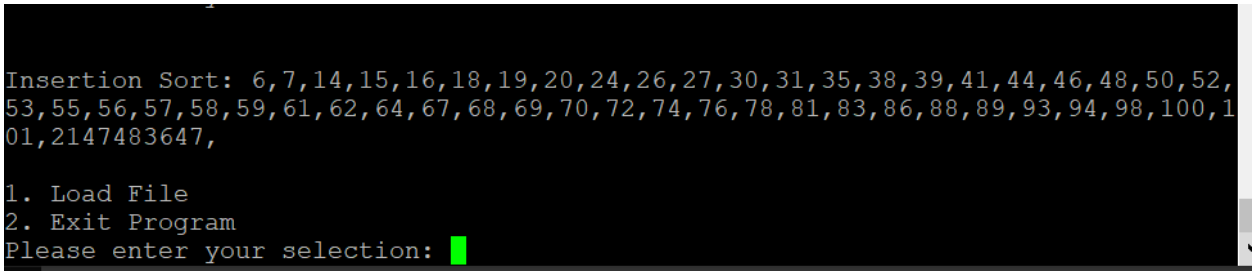
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**2147483648**,**101.5**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100**,101.5, 2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



<u>Test Case #026:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one larger than the maximum and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**2147483648, 450/2**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,81,83,86,88,89,93,94,98,100,4**50/2, 2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,4
50,2147483647,

1. Load File
```

Test Case #027:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one double and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,**101.5, 450/2**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,81,83,86,88,89,93,94,98,100,**101.5,450/2**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,1
01,450,

1. Load File
2. Exit Program
Please enter your selection:
```

Bubble Sort:

Traditional:

Test Case #000:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with no delimiter

Preconditions: N/A

Inputs:

391410016932462685276864815418355183074731446781702753596119563588587298386494695046786578926207949

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -1778461276,-1778461276,-1778461272,-1778461208,-1778461088,-177846
1072,0,0,0,0,0,0,37,32752,32752,32752,32752,32752,32752,32752,32752,32752,32752,
32752,32752,32752,32766,32766,32766,32766,32766,32766,32766,3167716,12721243,814
159578,1902851424,1906235964,1906358872,1906385680,1906385680,1906557660,1908560
240,1908565688,1908569476,1908569488,1908614012,1908615145,1908808432,2147483647
,

1. Load File
```

Test Case #001:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with greater than 50 elements

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49,**101**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**101**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
```

<u>Test Case #002:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with less than 50 elements

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,
55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,3
2532,
```

Test Case #003:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that uses letters as delimiters

Preconditions: N/A

Inputs:

39W14V100U16T93S24R62Q68P52O76n86M48L15K41J83I55H18G30F74E7D31C44B67A81Z70Y27

X53W59V61U19T56S35R88Q58P72O98N38M64L94K69J50I46H78G6F57E89D26C20B79A49

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69

,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -2036845216,-2033460676,-2033337768,-2033310960,-2033310960,-203313
8980,-2031136400,-2031130952,-2031127164,-2031127152,-2031082628,-2031081495,-20
30888208,-1825054300,-1825054300,-1825054296,-1825054232,-1825054112,-1825054096
,0,0,0,0,0,37,39,39,32620,32620,32620,32620,32620,32620,32620,32620,32620,32620,
32620,32620,32620,32764,32764,32764,32764,32764,32764,32764,3167716,12721243,814
159578,
```

Test Case #004:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers from a text file containing data on different lines

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30

,74,7,31,44,67,81,70,27,53,59,61,19,

56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89

,26,20,79,49

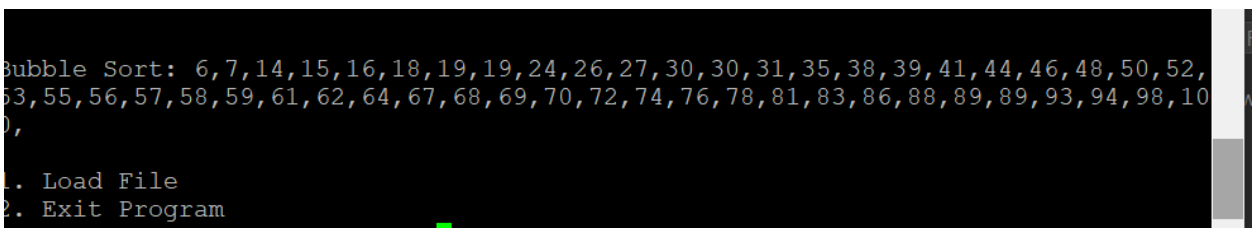
Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,53,55,56,57,58,59,61,62,64,67,68,69

,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Passed(10/9/24)



<u>Test Case #005:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values greater than the maximum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,**2147483648**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,79,81,83,86,88,89,93,94,98,100,**2147483648**

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,
55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,2
147483647,

1. Load File
2. Exit Program
```

Test Case #006:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values lower than the minimum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,-**2147483649**

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,
48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93
,94,98,100,

1. Load File
2. Exit Program
```

Test Case #007:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are doubles.

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,**79.32,49.12**

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
```

<u>Test Case #008:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are fractions

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,**7/3**,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49

Expected Output:

7/3,6,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
2. Exit Program
```

<u>Normal:</u>

<u>Test Case #009:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are just greater than the minimum value

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,-2147483648

Expected Output:

-2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Passed(10/9/24)

```
Please enter your selection: 2

Bubble Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,
48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93
,94,98,100,

1. Load File
```

Test Case #010:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are just smaller than the maximum value

Preconditions: N/A

Inputs:

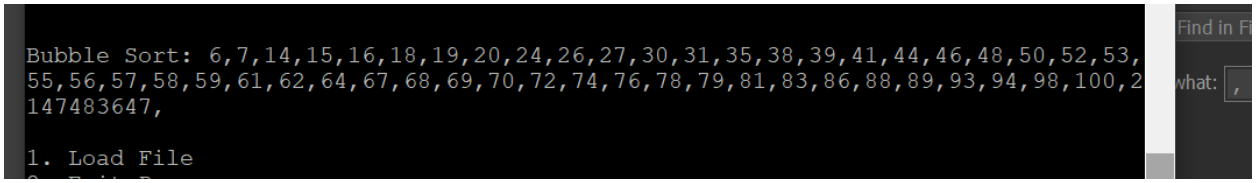
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,2147483647

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,2147483647

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)



Test Case #011:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that are all negative

Preconditions: N/A

Inputs:

-39,-14,-100,-16,-93,-24,-62,-68,-52,-76,-86,-48,-15,-41,-83,-55,-18,-30,-74,-7,-31,-44,-67,-81,-70,-27,-53,-59,-61,-19,-56,-35,-88,-58,-72,-98,-38,-64,-94,-69,-50,-46,-78,-6,-57,-89,-26,-20,-79,-49

Expected Output:

-100,-98,-94,-93,-89,-88,-86,-83,-81,-79,-78,-76,-74,-72,-70,-69,-68,-67,-64,-62,-61,-59,-58,-57,-56,-55,-53,-52,-50,-49,-48,-46,-44,-41,-39,-38,-35,-31,-30,-27,-26,-24,-20,-19,-18,-16,-15,-14,-7,-6,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Bubble Sort: -100,-98,-94,-93,-89,-88,-86,-83,-81,-79,-78,-76,-74,-72,-70,-69,-6
8,-67,-64,-62,-61,-59,-58,-57,-56,-55,-53,-52,-50,-49,-48,-46,-44,-41,-39,-38,-3
5,-31,-30,-27,-26,-24,-20,-19,-18,-16,-15,-14,-7,-6,

1. Load File
```

<u>Test Case #012:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has values that has repeated entries

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,89,79,79,79

Expected Output:

6,7,14,15,16,18,19,24,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,79,79,81,83,86,88,89,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,24,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,
57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,79,79,81,83,86,88,89,89,93,94,98,10
0,

1. Load File
```

<u>Weak Robust:</u>

<u>Test Case #013:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that alphabetical

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,A

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,5
3,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100
,

1. Load File
2. Exit Program
Please enter your selection:
```

<u>Test Case #014:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value greater than the maximum integer value in c++

Preconditions: N/A

Inputs:

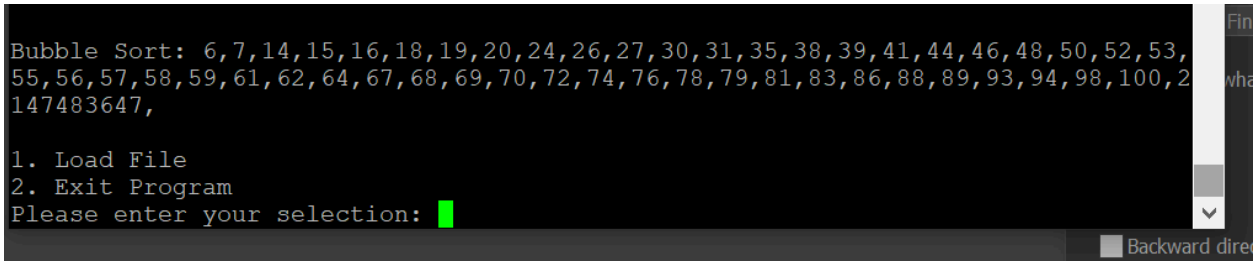
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,2147483648

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,2147483648

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



<u>Test Case #015:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value lower than the minimum integer value in c++

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,-2147483649

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,
48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93
,94,98,100,

1. Load File
2. Exit Program
Please enter your selection:
```

Test Case #016:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that is a double.

Preconditions: N/A

Inputs:39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49.12

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
2. Exit Program
Please enter your selection:
```

Test Case #017:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers that has a value that is a fraction

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7/3,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,49

Expected Output:

7/3,6,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49.12,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79.32,81,83,86,88,89,93,94,98,100

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,49,50,52,
53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
2. Exit Program
Please enter your selection:
```

Strong Robust:

Test Case #018:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one value smaller than the minimum.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,A,-2147483649

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
ubble Sort: -2147483648,0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,4
,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,
4,98,100,

. Load File
```

Test Case #019:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one larger than the maximum.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,A,2147483648

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 2147483648

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

Test Case #020:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one double.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,A,101.5

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 101.5

Expected Postconditions: Load File / Exit Program

```
Bubble Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,5
3,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,21
47483647,

1. Load File
```

Execution History: Failed (10/9/24)

<u>Test Case #021:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one alphabet character and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,A,300/2

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 300/2

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 0,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,5
3,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,30
0,

1. Load File
2. Exit Program
Please enter your selection:
```

<u>Test Case #022:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one larger than the maximum.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,-2147483649,2147483648

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 2147483648

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,
48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94
,98,100,2147483647,

1. Load File
2. Exit Program
```

Test Case #023:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one double.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,-2147483649,101.5

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 101.5

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -2147483648,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,
48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94
,98,100,101,
```

Test Case #024:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one smaller than the minimum and one fraction.

Preconditions: N/A

Inputs:

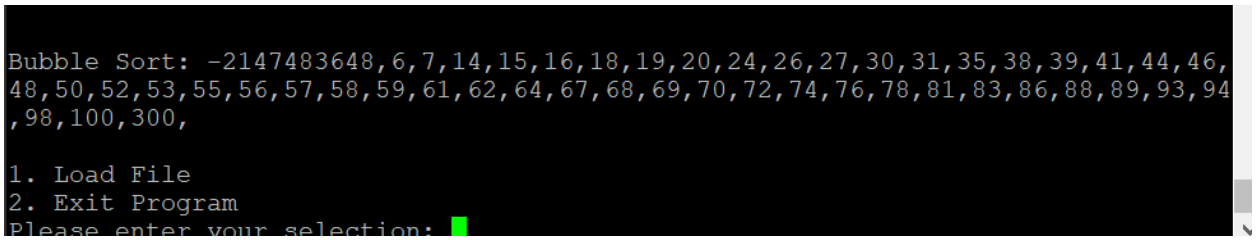
39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,-2147483649, 300/2

Expected Output:

-2147483649,6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100, 300/2

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)



<u>Test Case #025:</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one larger than the maximum and one double.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,2147483648,101.5

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,101.5, 2147483648

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,
55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,101,
2147483647,

1. Load File
```

Test Case #026:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one larger than the maximum and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,

72,98,38,64,94,69,50,46,78,6,57,89,26,20,2147483648, 450/2

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70

,72,74,76,78,81,83,86,88,89,93,94,98,100,450/2, 2147483648

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,
55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,450,
2147483647,

1. Load File
```

Test Case #027:

Purpose: The purpose of this test is to demonstrate that the program correctly sorts a list of integers with one double and one fraction.

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,18,30,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,101.5, 450/2

Expected Output:

6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,101.5,450/2

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,14,15,16,18,19,20,24,26,27,30,31,35,38,39,41,44,46,48,50,52,53,
55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,81,83,86,88,89,93,94,98,100,101,
450,

1. Load File
2. Exit Program
```

Decision Table-Based Testing

Bubble Sort

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Negatives	T	T	T	T	F	F	F	F
Doubles	T	T	F	F	T	T	F	F
Characters	T	F	T	F	T	F	T	F
Output	✗	✗	✗	✓	✗	✗	✗	✓

Insertion Sort

Conditions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Negatives	T	T	T	T	F	F	F	F
Doubles	T	T	F	F	T	T	F	F
Characters	T	F	T	F	T	F	T	F
Output	✗	✗	✗	✓	✗	✗	✗	✓

Test Case #0

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for doubles

Preconditions: N/A

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,

70,72,74,76,78,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,5
3,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100
,

1. Load File
2. Exit Program
```

Test Case #1

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for doubles

Preconditions: Test case 0

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58

,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,

70,72,74,76,78,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,5
2,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,
100,

1. Load File
2. Exit Program
```

Test Case #2

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for doubles and characters

Preconditions: Test case 1

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,A,6,57,89,26,20,79,17,

Expected Output:

A,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52
,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,79,81,83,86,88,89,93,94,98,100,

1. Load File
2. Exit Program
```

Test Case #3

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for doubles and characters

Preconditions: Test case 2

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,A,6,57,89,26,20,79,17,

Expected Output:

A,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50
,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,79,81,83,86,88,89,93,94,98,1
00,

1. Load File
2. Exit Program
```

Test Case #4

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for doubles, characters and negatives

Preconditions: Test case #3

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,A,6,57,89,26,20,79,17,

Expected Output:

A,-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

```
Bubble Sort: -70,0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,5
0,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,79,81,83,86,88,89,93,94,98,100
,

1. Load File
2. Exit Program
```

Execution History: Failed (10/9/24)

Test Case #5

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for doubles, characters and negatives

Preconditions: Test case #4

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,A,6,57,89,26,20,79,17,

Expected Output:

A,-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -70,0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,4
8,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,79,81,83,86,88,89,93,94,98,
100,

1. Load File
2. Exit Program
```

<u>Test Case #6</u>

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for doubles and negatives

Preconditions: Test case 5

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
2. Exit Program
```

Test Case #7

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for doubles and negatives

Preconditions: Test case 6

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83.5,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83.5,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,
50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,

1. Load File
2. Exit Program
```

Test Case #8

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for characters

Preconditions: Test case 7

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,70,27,A,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

A,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: 0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52
,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

1. Load File
2. Exit Program
```

Test Case #9

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for characters

Preconditions: Test case 8

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,70,27,A,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

A,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: 0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50
,52,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,1
00,

1. Load File
2. Exit Program
```

Test Case #10

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for characters and negatives

Preconditions: Test case 9

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,-70,27,A,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

A,-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Bubble Sort: -70,0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,5
0,52,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,100
,

1. Load File
2. Exit Program
```

Test Case #11

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for characters and negatives

Preconditions: Test case 10

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,-70,27,A,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

A,-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Failed (10/9/24)

```
Insertion Sort: -70,0,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,4
8,50,52,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,
100,

1. Load File
2. Exit Program
```

Test Case #12

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with tests for negatives

Preconditions: Test case 11

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Bubble Sort: -70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,
52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,10
0,

1. Load File
2. Exit Program
```

Test Case #13

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using insertion sort with tests for negatives

Preconditions: Test case 12

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,-70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

-70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Insertion Sort: -70,6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,
50,52,53,55,56,57,58,59,61,62,64,67,68,69,72,74,76,78,79,81,83,86,88,89,93,94,98
,100,

1. Load File
2. Exit Program
```

Test Case #14

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with all standard inputs

Preconditions: Test case 13

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Bubble Sort: 6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,5
3,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100
,

1. Load File
2. Exit Program
```

Test Case #15

Purpose: The purpose of this test is to demonstrate that the program correctly sorts the array using bubble sort with all standard inputs

Preconditions: Test case 14

Inputs:

39,14,100,16,93,24,62,68,52,76,86,48,15,41,83,55,8,33,74,7,31,44,67,81,70,27,53,59,61,19,56,35,88,58,72,98,38,64,94,69,50,46,78,6,57,89,26,20,79,17,

Expected Output:

6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,52,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,100,

Expected Postconditions: Load File / Exit Program

Execution History: Passed (10/9/24)

```
Insertion Sort: 6,7,8,14,15,16,17,19,20,24,26,27,31,33,35,38,39,41,44,46,48,50,5
2,53,55,56,57,58,59,61,62,64,67,68,69,70,72,74,76,78,79,81,83,86,88,89,93,94,98,
100,

1. Load File
2. Exit Program
```

Reflections:

Sean;

I was in charge of the Decision Table-Based Testing. I found the hardest part of decision based testing to be determining effective conditions. I had run a few basic tests at the beginning to see what conditions would be most worthwhile to do a deeper dive. I initially tested very large integers, very small integers, doubles, negatives, characters, no spaces, and multiple of the same letter. I also tested the UI by trying to select invalid options such as 3 and -1 but the program would correctly re-prompt the user for an input and there was no more testing needed for that aspect of the program. The program performed well within the specifications so I decided to focus on invalid entries. After deciding to test doubles, characters, and negatives I created a truth table to organize all the different tests with their corresponding results. The program was able to consistently sort the negative numbers but it was unable to handle doubles and characters (since they were out of the specifications), the program would truncate the doubles to be integers and would replace the characters with a 0, after converting the inputs to integers it was able to correctly sort them.

Aman:

I was in charge of some of the equivalence class testing. I performed manual testing and screenshotting the results as well as making sure to modify the values based on the type of equivalence testing we did. Another one of the main aspects of this project was figuring out how to make all the entires in the sorting algorithm problems

Gavin:

I was in charge of the equivalence class testing. The hardest part of equivalence class testing was determining which element to pick out and use as a test case. In this case, as it is a sorting algorithm, entries in the unsorted array are redundant, as insertion and bubble sort perform the same operations on

each item in the list. For this reason, a test case with one variable changed, say in the 40th position, would be equivalent to the same change but in any other position in the unsorted list. This simplified things, and for traditional testing it ensured that only one item needed to be changed to determine the validity of the test for all of the elements in the array. I determined all of the invalid inputs through initial testing in which I was trying to break the program. I incorrectly assumed that a list spanning different lines would cause the program to malfunction, but it did not. Every other test case I deemed as invalid caused an incorrect result, and the ones determined through traditional and weak robust testing were as follows: a list with no delimiter, greater than / less than 50 elements, using letters as delimiters, an entry with a smaller than minimum value / greater than maximum value, doubles, fractions, and an alphabet character as an integer. For strong robust testing, as we had already determined all of the valid boundary values through normal testing, we only needed to test each invalid value against each other. The program always produced an output - it never 'broke' - but things such as an integer larger than the maximum value caused the program to round the value back down to the maximum value. The actual sorting algorithm was not determined to be faulty by our tests - rather how the program passed the values from the text file. Even in Test Case #003, in which the introduction of letters completely broke the values inserted into the array, they were still sorted correctly according to their value. The program truncated double values, as well as fractions. Additionally, it set alphabetical characters to 0. Initially, I tested all the cases on insertion sort, and then reused the test cases for bubble sort. All of the results were the same, which is why I came to the conclusion that the issue was not with either sorting algorithm but rather how the program read in information from the file.