Algorithms in C++: Assignment 5

1. Objective

Your goal is to write a program that will enable you to make use of mergesort to solve a seemingly unrelated problem. [This is a common interview question. I too have been asked this one!]

2. Problem

You have been asked to find the number of inversions in an array. An inversion is when a larger number appears before a smaller number. In the follow example, there are 3 inversions.

```
[3, 2, 1]
1. 3 before 2
2. 3 before 1
3. 2 before 1
```

You need to write two different algorithms to solve this problem. One is "slow". It is the naïve approach using nested loops. The "fast" approach uses a modified mergesort that counts the number of inversions and returns that count. Your program will always run the fast algorithm unless the user specifies "slow" as the (only) command-line argument.

Here are some examples of how the program should run:

```
$ ./inversioncounter
Enter sequence of integers, each followed by a space: x 1 2 3
Error: Non-integer value 'x' received at index 0.

$ ./inversioncounter
Enter sequence of integers, each followed by a space: <some spaces>
Error: Sequence of integers not received.

$ ./inversioncounter slow
Enter sequence of integers, each followed by a space: 1 1 1 1
Number of inversions: 0

$ ./inversioncounter
Enter sequence of integers, each followed by a space: 3 1 0 1 2 9
Number of inversions: 5
```

3. Advice

We store the input in a vector, since we don't know how many values the user will enter. You can easily pass the internal array of the vector to the function with `&values[0]`.

Make sure you test your program on large inputs. Test cases 17-20 and 33-36 have 100,000 numbers.

You are allowed up to 8 seconds on the Lubuntu virtual machine for the "slow" version and 1.25 seconds for the "fast" version. You should have no trouble meeting these timing requirements.

Use the template file provided.

Important: Change the -g in the makefile to be -O3 (that's a capital O) when you are ready to test your code for speed. This flag tells the compiler to optimize the code it generates, which will enable you to meet the time constraints imposed by the test script.