```
Author: Christopher D. Coleman
CS 315-002
Homework 9
Version 1.0.1

COMPILING IN LINUX
------------------
This program was created in Visual Studio 2012/2015, but if you would like to compile in linux follow these
instructions:
1. If you transfer the all the files straight to a linux based computer, then compile with sourceLINUX.cpp instead
of source.cpp
2a. g++ -std=c++11 -o HW9 *.cpp
          or
2b. g++ -std=c++11 -o HW9 sourceLINUX.cpp Menu.cpp Sorter.cpp

Make sure if you are using the above command (2a) that you do not have source.cpp downloaded in the same folder,
you want
to compile with sourceLINUX.cpp

Then to excute use the command: ./HW9

INSTRUCTIONS:
  The program will prompt the user with the choices:
    a: generate 20 sequences and find the average comparisons.
    r: generate random sequence of n size.
    g: to generate random sequences of sizes going to n
    c: create a sequence with integers given by user.
    p: print the sequence of integers.
    i: use the insertion sort method to sort the sequence of integers.
    m: use the merge sort method to sort the sequence of integers.
    s: use the selection sort method to sort the sequence of integers.
    q: use the quick sort method to sort the sequence of integers.

There are measures in place to catch most input exceptions, but to make sure there is no errors please try to only
enter
valid input.

======================
PROGRAM DESCRIPTION:
  This program allows the user to input a random sequence of integers and allows them to sort
  the sequence using either insertion sort, selection sort, quicksort (with the random selection of pivot)
  and mergesort to sort a vector.

-------------------
CLASS: Sorter
===================
FUNCTIONS:
  Sorter(int size):
    Constructor that takes an int that is the size of the sequence.

  Sorter():
    Default constructor initializes vector<int> seq to size = 11.

  void print():
    Prints every element in vector<int> seq.

  void insertionSort():
    Sorts a sequence of integers as a vector<int> by using the insertion sort method.
    Average runtime O(n^2).

  void selectionSort():
    Sorts a sequence of integers as a vector<int> by using the selection sort method.
    Average runtime O(n^2)

  void quickSort(int left, int right):
    Sorts a sequence of integers as a vector<int> by using the quick sort method.
    Average runtime O(n)

  void merge(int l, int m, int r):
    A function that comes along with merge sort to "combine" segments of the sequence.

  void mergeSort(int l, int r):
    Sorts a sequence of integers as a vector<int> by using the merge sort method.
    Average runtime O(n)

  vector<int>* getSeq():
    Returns the sequence of numbers to be sorted.

  int get_size():
    Returns the size of the sequence of numbers being sorted.
  int get_counter():
    Returns int counter, the number of comparisons made.

  void printLess():
    Prints the number of comparisons that the previous sort took.

PRIVATE:
  vector<int> seq: a sequence of integers to be sorted.
  int counter: counts the number of comparisons each sorting algorithm uses.

-----------------
CLASS: Menu
=================
FUNCTIONS:
  Menu(int size):
    Constructor that takes a int and creates a Sorter object with the size of the (int) given.

  Menu():
    Default constructor that creates a Sorter object with the size of 32.

  void printMenu():
    Displays a menu with the commands you can use in this program.

  void setChoice():
    Sets the variable choice to whatever the user inputs.

  void processChoice():
    This functions holds a switch function that handles the users input and executes accordingly.

  void createSequence():
    Allows the user to create a sequence of ints, up to the size of their choosing.

  void generateRandomSeq():
    Creates a random sequence of numbers and the user decides the size of the sequence.

  void generateRandomSeqTo():
    This is a function used for analysing the comparisons each sort take. The function will
    allow you to enter a number and the program will create that many random sequences.
    Such as if the user inputs 6400, the program will create 6400 random sequences and sort them.

  void averageComparisons():
    Generates the average comparisons each sorting method takes for n = {100, 200, 400, 800, 1600, 3200, 6400}.

PRIVATE:
  char choice: The command the user inputs.
  Sorter *sort: The Sorter object that will handle the sequences of integers.
```