CMP1001M
Software Development
Assessment 1
Text Analysis Program

Ryan Docherty

Contents

Implemented/Extra Options

The options that I have implemented are as follows:

The number of sentences, vowels, consonants, upper-case letters, lower-case letters, frequency of individual letters and words longer than seven letters are outputted to a file.

The extra option that I have implemented is the number of individual digits.

Test Plan Analysis

My text analysis program contains four classes and roughly eight different methods. Therefore, I will be carrying out a series of black box testing and white box testing to ensure that they are fully functional and any errors produced are dealt with appropriately.

Black Box Testing

As part of my black box testing, I will input only two different values into each method, considering that my program has no boundary-specific requirements (for example: 'Enter a value between 1 and 9'). The different values include an expected value and an erroneous value. Alongside these values, I will make note of the purpose of the test, expected output, the actual output, errors found and any error handling that I have added to my code.

White Box Testing

As part of my white box testing, I will input different values into each method like black box testing, but this time, I will put temporary Console.WriteLine statements which confirm that my program is entering/using the correct methods. For example, when I test my LetterFrequency method, I will put a temporary Console.WriteLine every time the if statement is used saying “Currently in the if statement, this method works” or something similar. I will also be making note of the purpose of the test, expected output, the actual output, errors found and any error handling that I have added to my code.

Methods That I will be Testing

Here are the methods that I will be testing:

NumOfSentences()
NumOfVowels()
NumOfConsonants()
NumOfUpperCase()
NumOfLowerCase()
NumOfDigits()
LetterFrequency()
LongWords()

These are the main methods that return values, but I will also be testing Main(). Below I will concisely explain each method's function, and possible valid and invalid data examples that may form part of my testing.

Test Plan – Main()

This method displays information on the program, as well as prompting what to enter depending on whether they want to enter via a keyboard or read from a text file.

Valid data: 'k' or 'K' for keyboard and 'f' or 'F' for file.
Invalid/Erroneous data: anything that isn't from above

Test Plan – NumOfSentences()

This method counts certain characters that are punctuation marks to determine the number of sentences. It then returns an integer as its value.

Valid data: these three punctuation marks - '.', '?', '!'
Invalid/Erroneous data: anything that isn't from above - '#', '&', '5' etc.

Test Plan – NumOfVowels()

This method counts certain characters that are vowels. Upper and lower case letters are valid. It then returns an integer as its value.

Valid Data: upper/lower case vowels
Invalid/Erroneous Data: anything that isn't from above – 'q', '5', '#' etc.

Test Plan – NumOfConsonants()

This method counts certain characters that are consonants. Upper case and lower case letters are valid. It then returns an integer as its value.

Valid data: upper/lower case consonants
Invalid/Erroneous data: anything that isn't from above 'a', '5', '#' etc.

Test Plan – NumOfUpperCase()

This method counts the number of upper case letters. It returns an integer as its value.

Valid data: any upper case letter
Invalid/Erroneous data: anything that isn't from above – 'a', '5', '#' etc.

Test Plan – NumOfLowerCase()

This method counts the number of lower case letters. It returns an integer as its value.

Valid data: any lower case letter
Invalid/Erroneous data: anything that isn't from above – 'A', '5', '#' etc.

Test Plan – NumOfDigits()

This method counts the number of digits. It return an integer as its value.

Valid data: any single digit
Invalid/Erroneous data: anything that isn't from above – 'a', 'A', '#' etc.

Test Plan – LetterFrequency()

This method loops through every character and stores each letter found in an array. It returns multiple integers as its value which correspond to the individual letters.

Valid data: any letter
Invalid/Erroneous data: anything that isn't from above – '?', '#', '1' etc.

Test Plan – LongWords()

This method splits the string entered into sub-strings every time a space is found, then these sub-strings are stripped of punctuation. The final sub-strings are then appended to a file if their length of letters is above seven.

Valid data: any string containing letters
Invalid/Erroneous data: anything that isn't from above – '5', '£$%#&' etc.

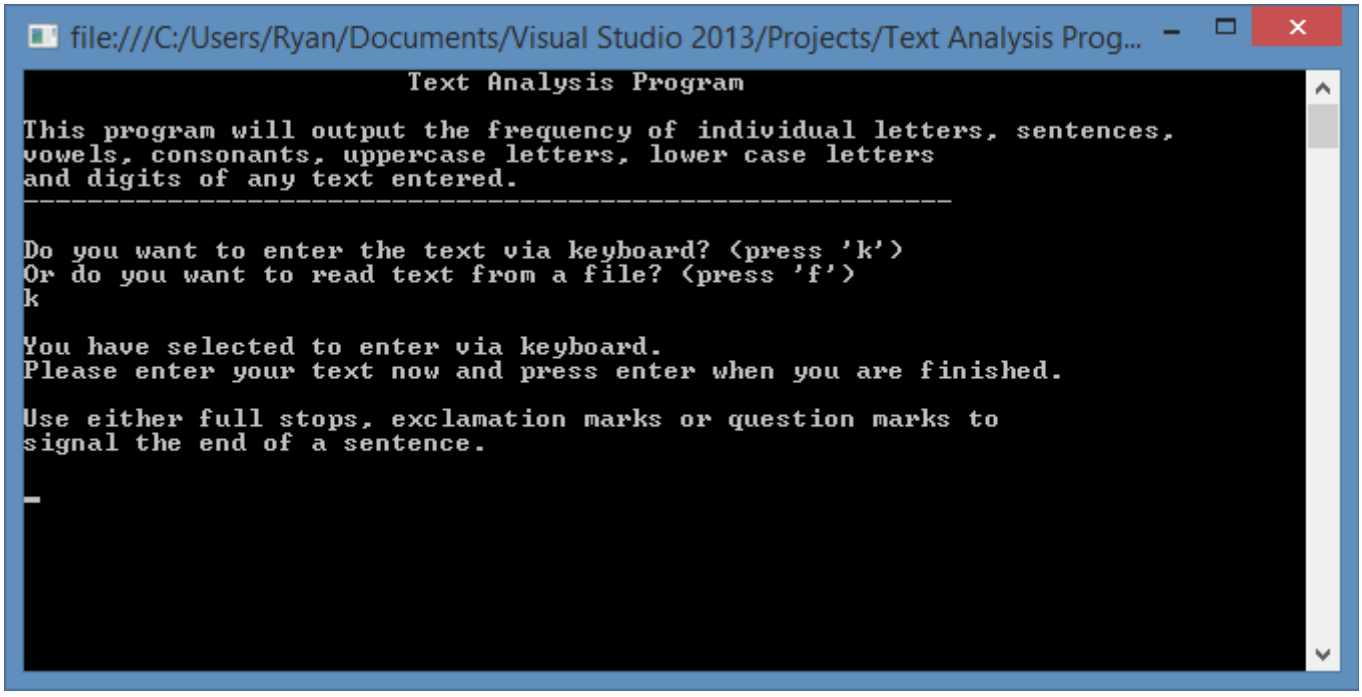
Black Box Testing

In this section, I will create tables and demonstrate the various inputs that I talked about earlier. These inputs will contain a normal/expected value and an unexpected/erroneous value.

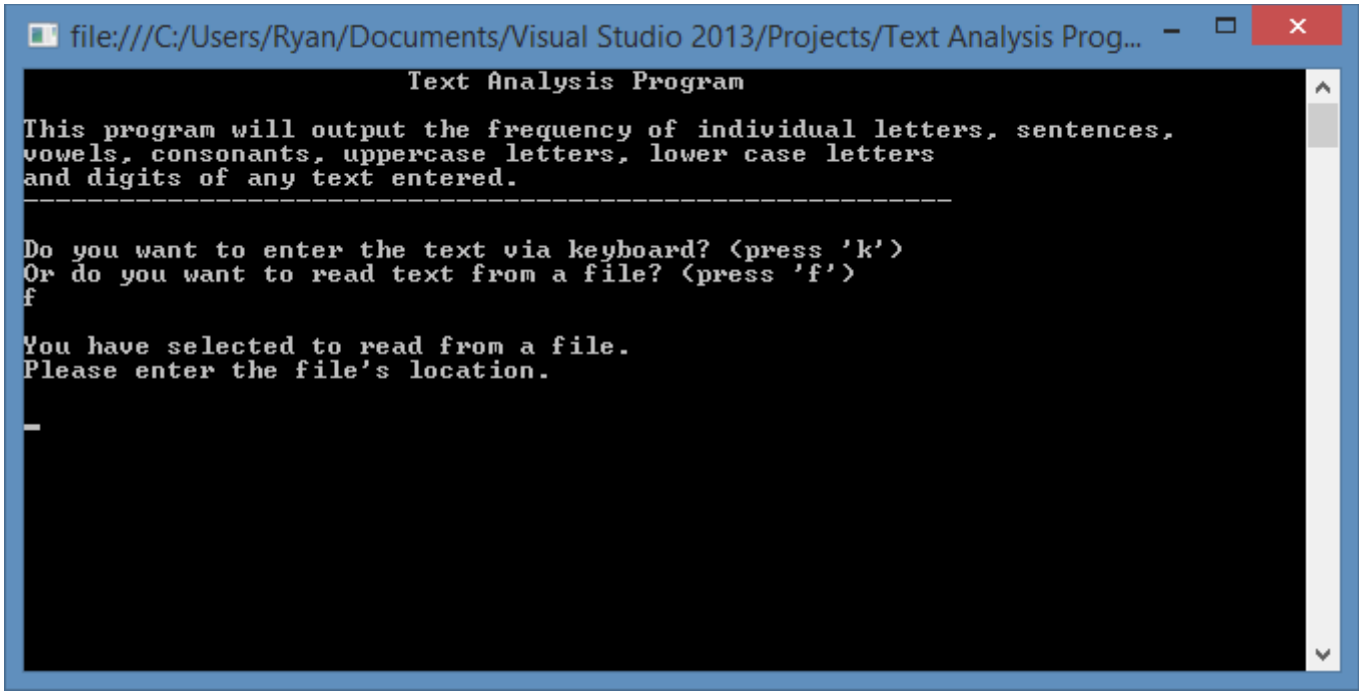
Black Box Testing – Main()

For this test, I will enter a lower and upper case 'K' and 'F' as the expected input and then any other input as an unexpected/erroneous input. I will also text the exception handling for this method.

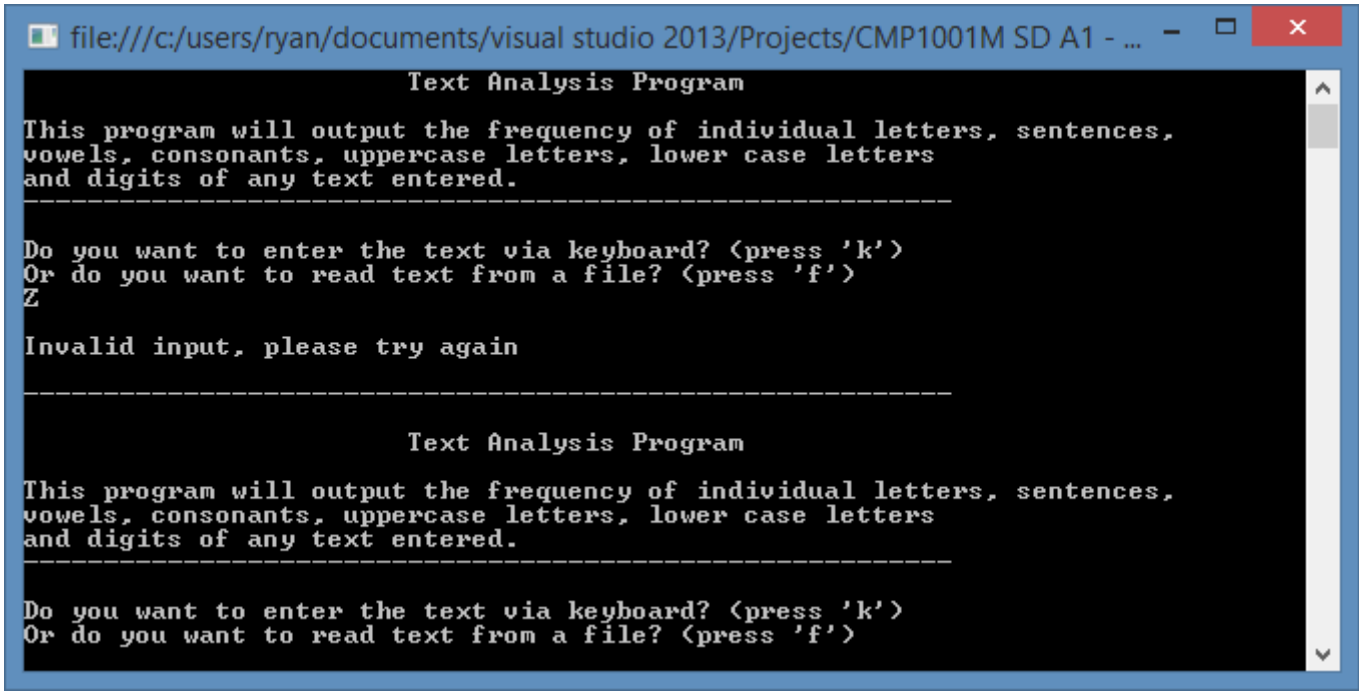
Test number	Purpose	Input	Expected Result	Actual Result
1	To see if this method outputs the correct text and calls other methods based on the user input	'k' or 'K'	Display message stating that I have selected to enter via keyboard	As expected



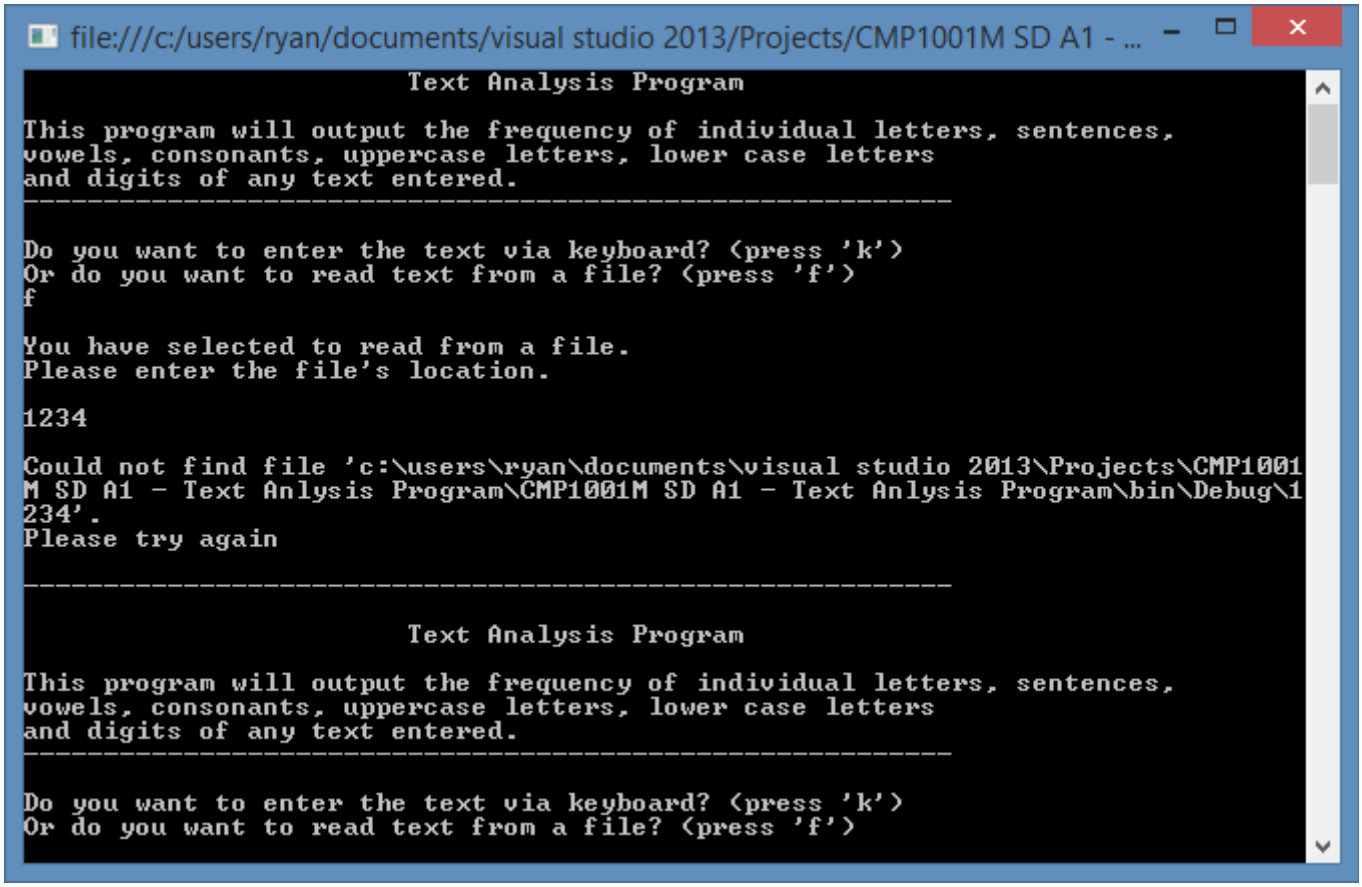
Test number	Purpose	Input	Expected Result	Actual Result
2	To see if this method outputs the correct text and calls other methods based on the user input	'f' or 'F'	Display message stating that I have selected to enter via a file	As expected



Test number	Purpose	Input	Expected Result	Actual Result
3	To see that this method doesn't accept other letters as an input	'Z'	Error message Main() method called	As expected



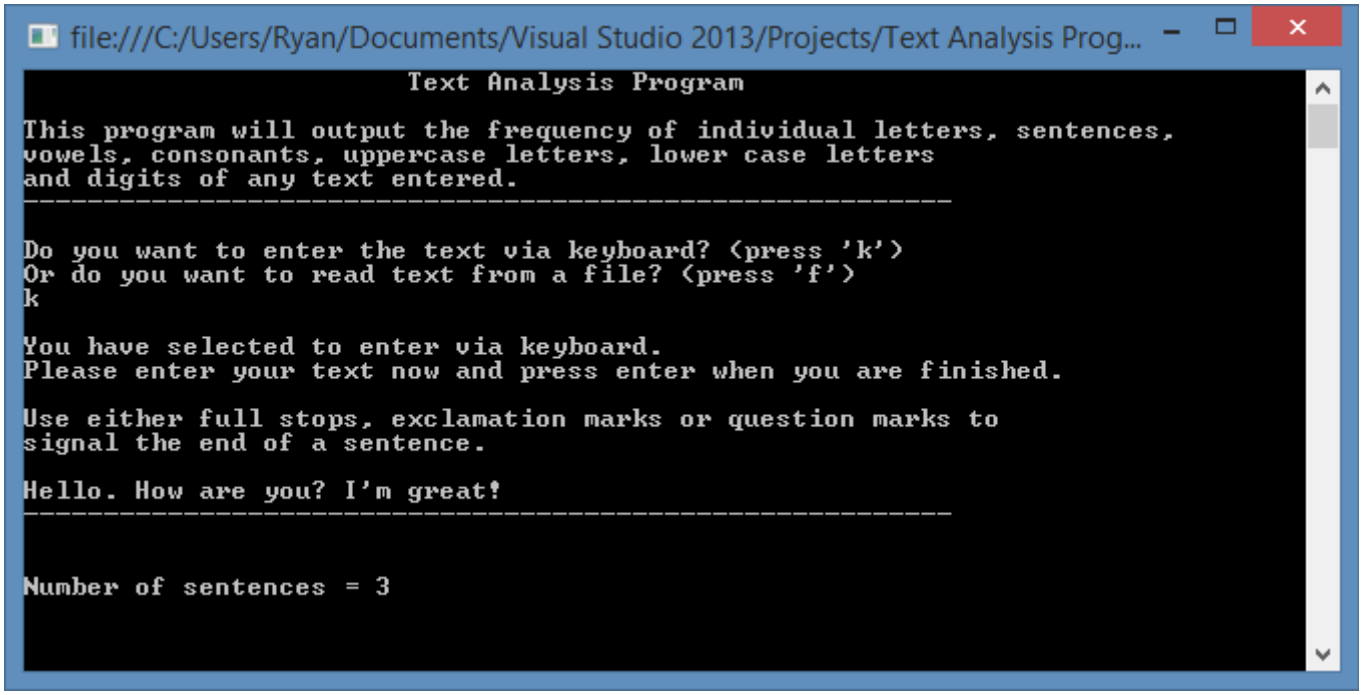
Test number	Purpose	Input	Expected Result	Actual Result
4	To see if the exception handling in this method works	'1234'	Error message Main() method called	As expected



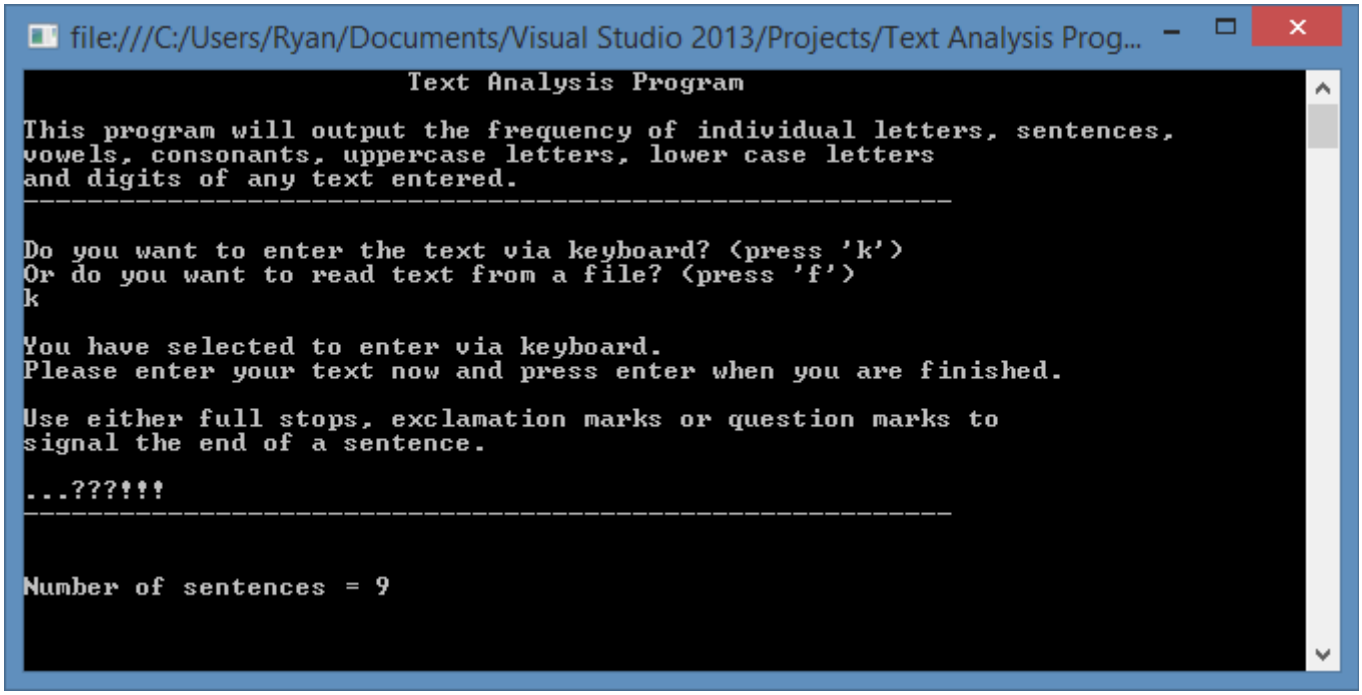
Black Box Testing - NumOfSentences()

For this test, I will enter each three punctuation marks ('.', '?' and '!') then I will enter an unexpected character.

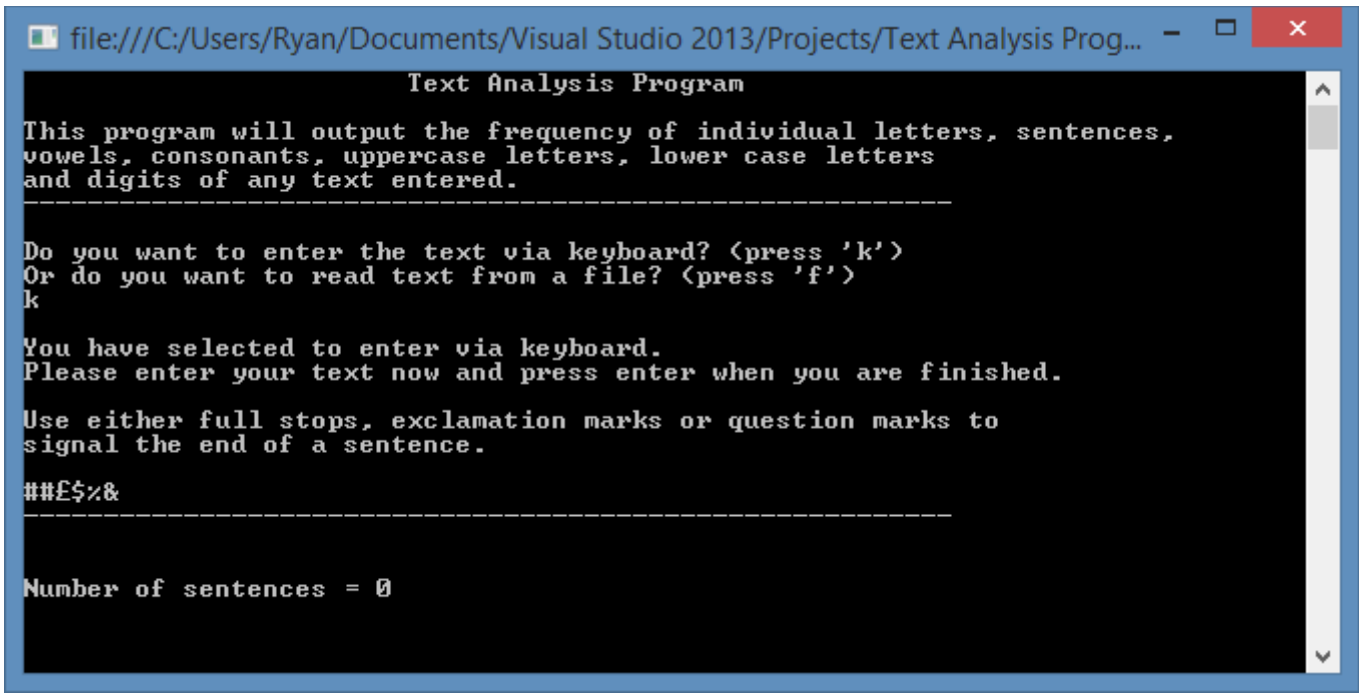
Test number	Purpose	Input	Expected Result	Actual Result
5	To see if this method counts the number of punctuation marks correctly in a normal series of sentences	“Hello. How are you? I'm great!”	Number of sentences = 3	As expected



Test number	Purpose	Input	Expected Result	Actual Result
6	To see if this method only counts punctuation marks	"...???!!!"	Number of sentences = 9	As expected



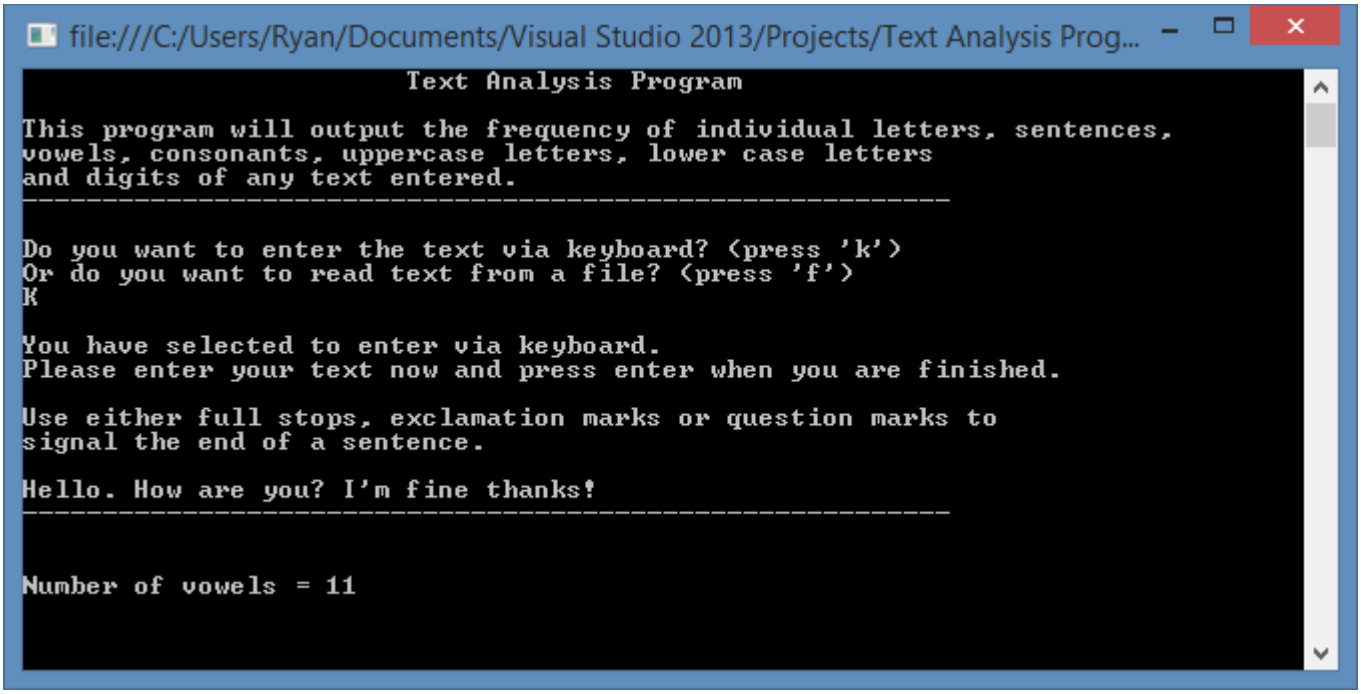
Test number	Purpose	Input	Expected Result	Actual Result
7	To see if this method only counts punctuation marks	"##£$%&"	Number of sentences = 0	As expected



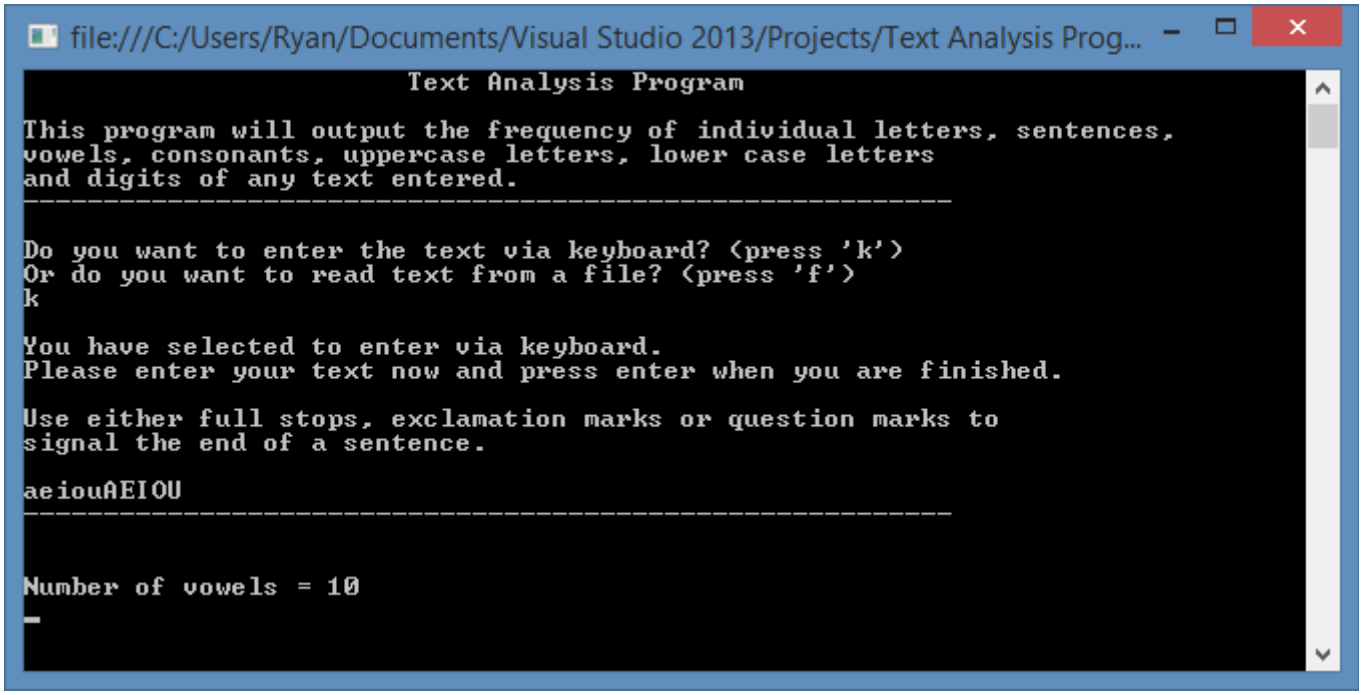
Black Box Testing – NumOfVowels()

For this test, I will enter a normal sentence with both vowels and consonants, then just vowels and finally just consonants/other characters.

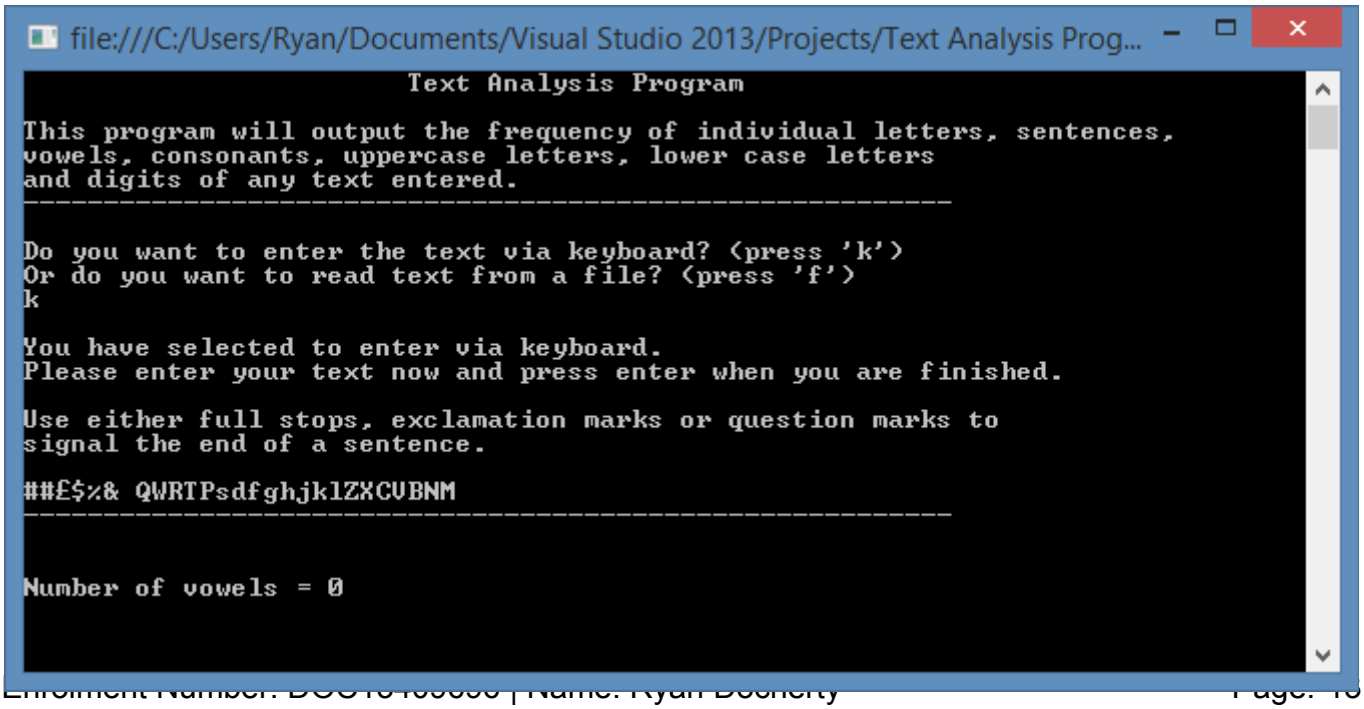
Test number	Purpose	Input	Expected Result	Actual Result
8	To see if this method only counts vowels correctly in a normal series of sentences	“Hello. How are you? I'm fine thanks!”	Number of vowels = 11	As expected



Test number	Purpose	Input	Expected Result	Actual Result
9	To see if this method counts all vowels	“aeiouAEIOU”	Number of vowels = 10	As expected



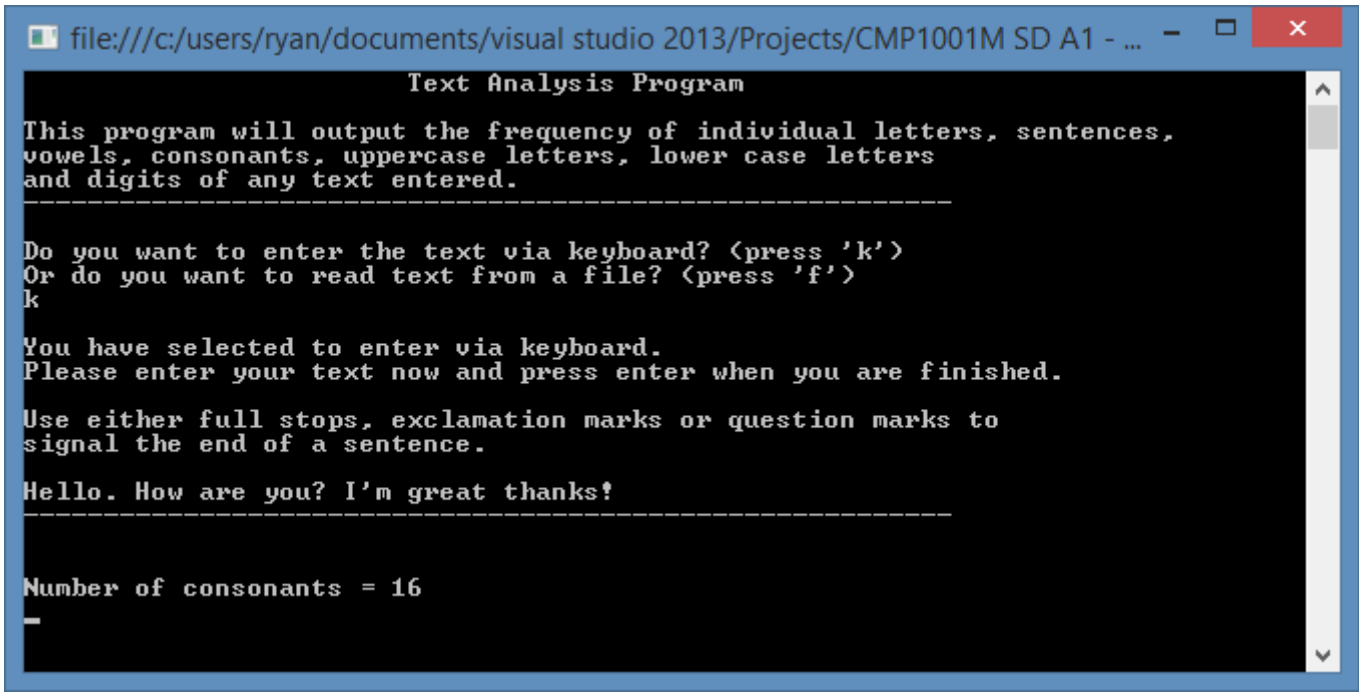
Test number	Purpose	Input	Expected Result	Actual Result
10	To verify that this method doesn't count characters that aren't vowels	"##£$%& QWRTPsdfghjkl ZXCVBNM"	Number of vowels = 0	As expected



Black Box Testing – NumOfConsonants()

For this test, I will enter a normal sentence with both vowels and consonants, then just consonants and finally just vowels/other characters.

Test number	Purpose	Input	Expected Result	Actual Result
11	To see if this method only counts consonants correctly in a normal series of sentences	“Hello. How are you? I'm great thanks!”	Number of consonants = 16	As expected

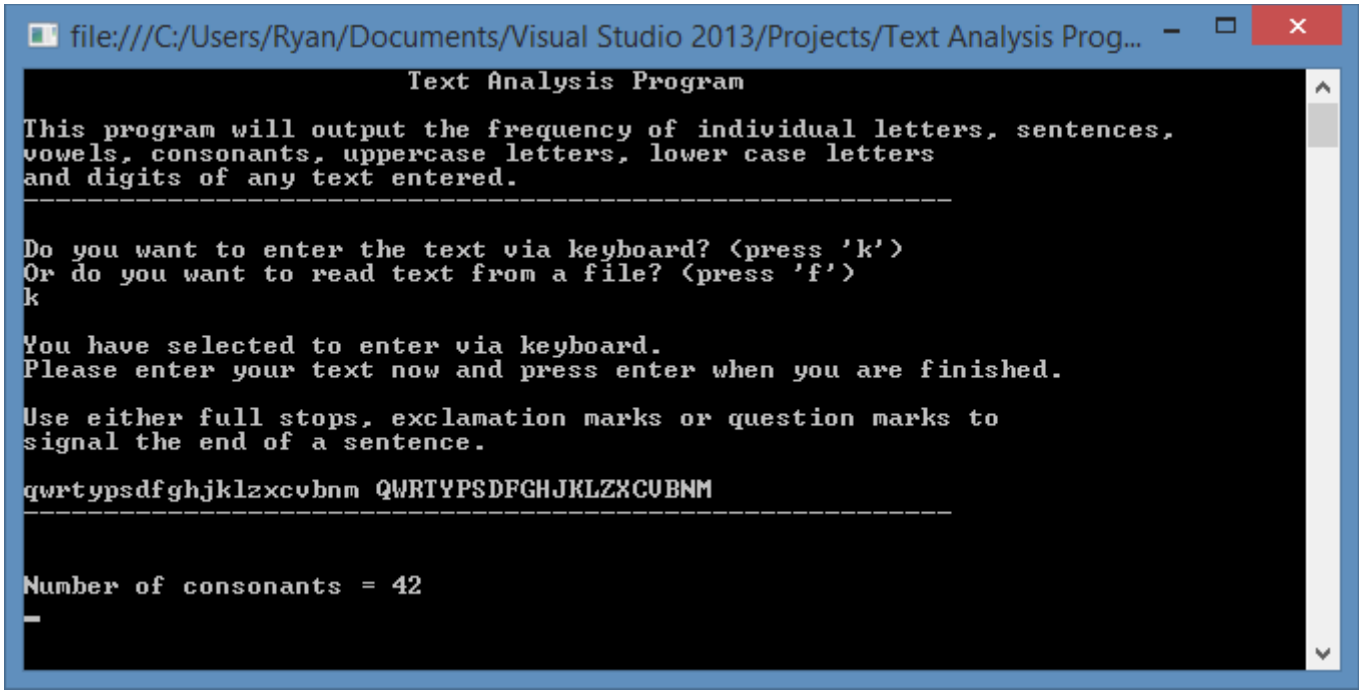


Test number	Purpose	Input	Expected Result	Actual Result
12	To see if this method counts all consonants	“qwrtypsdfghjkl zxcvbnm QWRTYPSDFG HJKLZXCVBN M”	Number of consonants = 42	Number of consonants = 39

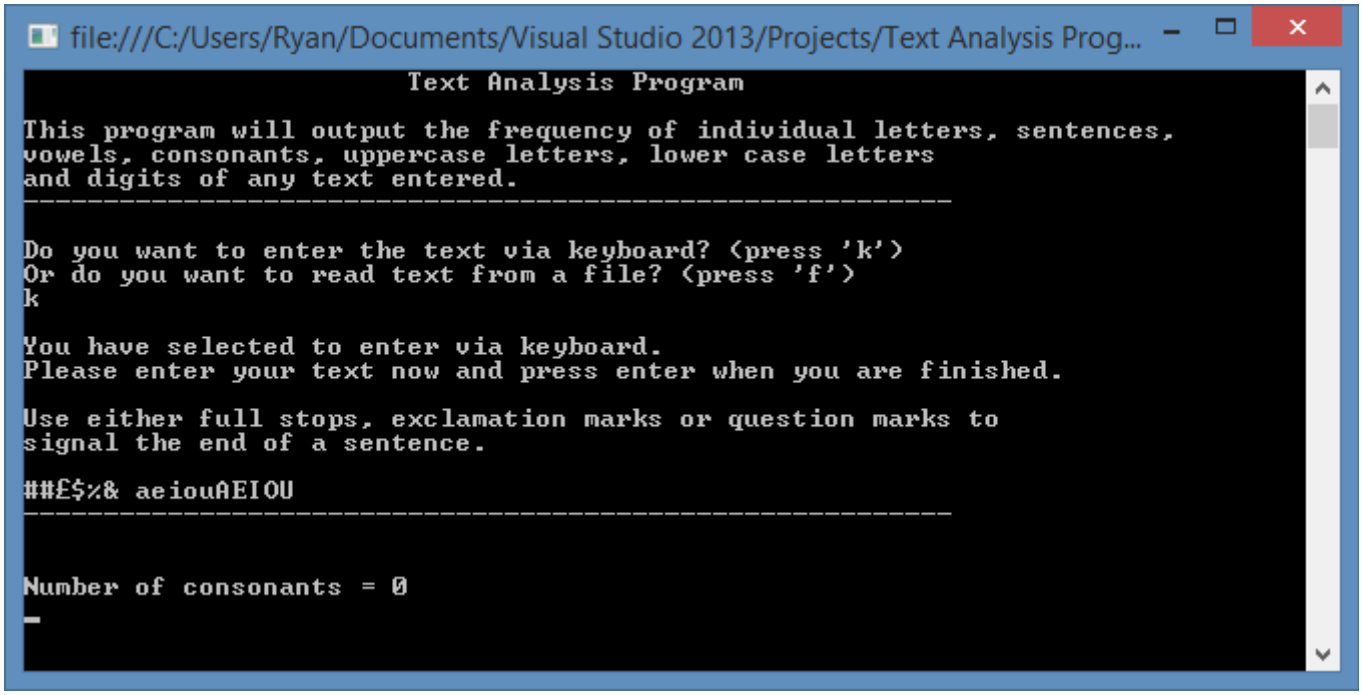
As an unexpected error occurred, I put a breakpoint at the start of this method and ran though the code line by line. The error that I found is a logical error in that I haven't defined the consonants correctly; three of the letters were missing, so the variable that

was holding the number of consonants wasn't incrementing. After correcting the code, I ran it again.

Test number	Purpose	Input	Expected Result	Actual Result
13	To see if this method counts all consonants	“qwrtypsdfghjkl zxcvbnm QWRTYPSDFG HJKLZXCVBN M”	Number of consonants = 42	As expected



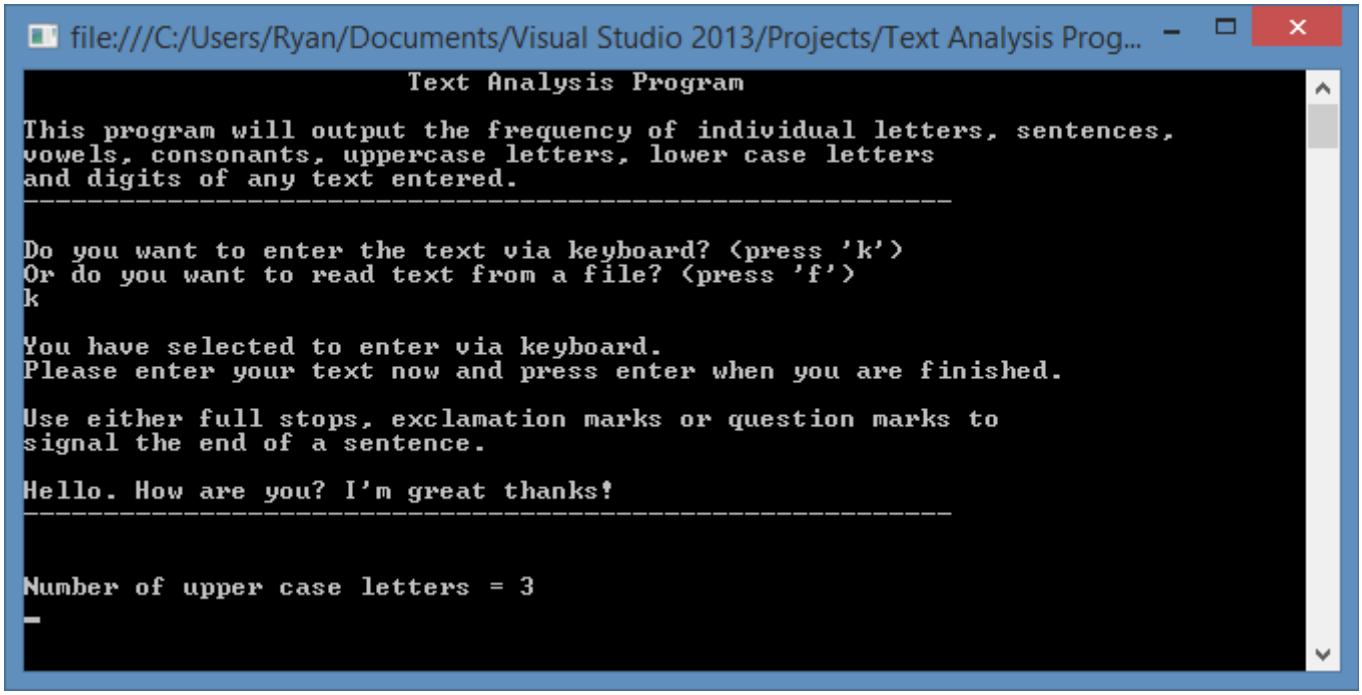
Test number	Purpose	Input	Expected Result	Actual Result
14	To verify that this method doesn't count characters that aren't consonants	“##£$%& aeiouAEIOU”	Number of consonants = 0	As expected



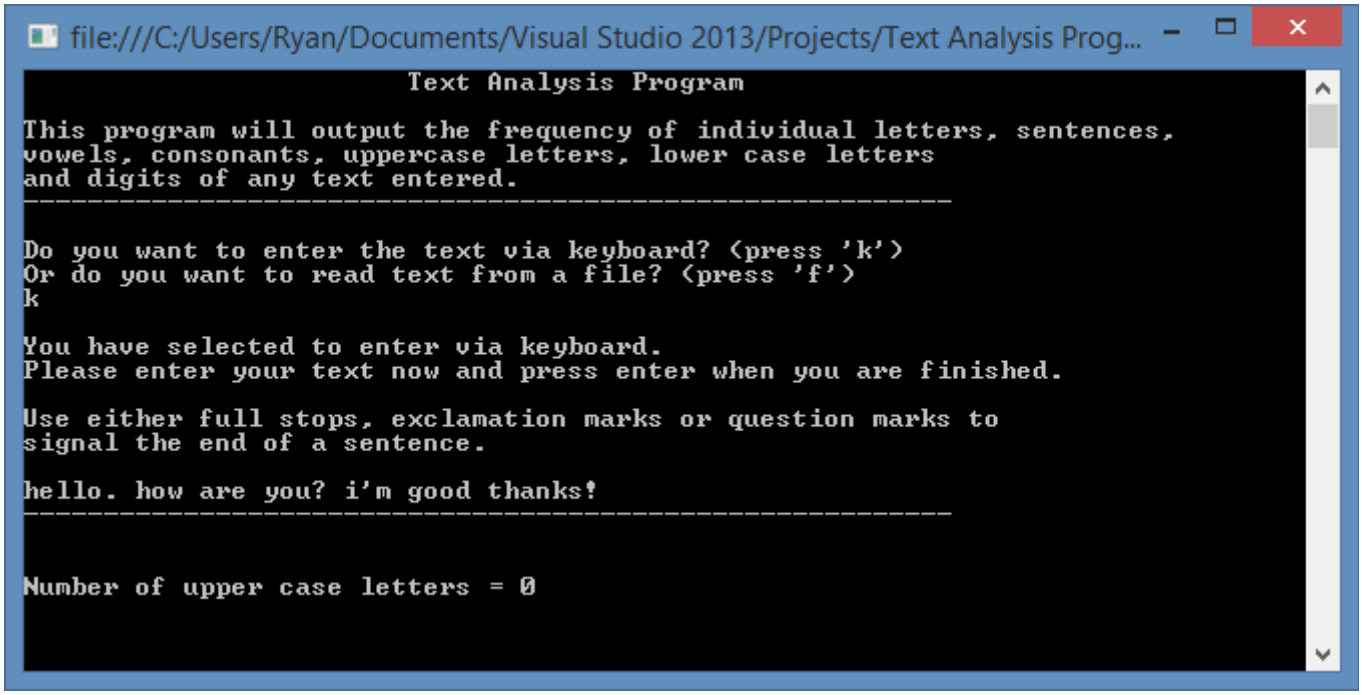
Black Box Testing – NumOfUpperCase()

For this test, I will enter a normal/expected sentence with a mix of upper-case and lower-case letters, then just upper-case letters and then finally just lower-case/other invalid characters.

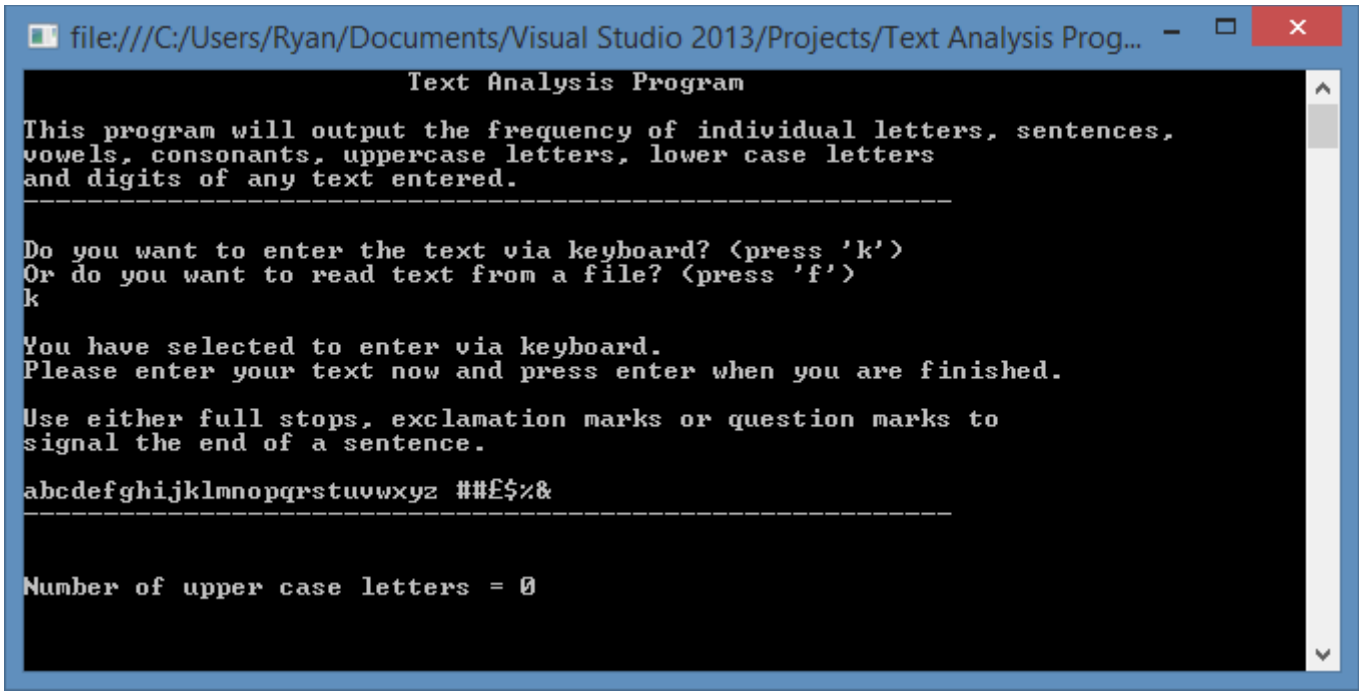
Test number	Purpose	Input	Expected Result	Actual Result
15	To verify that this method only counts upper-case letters correctly in a normal series of sentences	“Hello. How are you? I'm great thanks!”	Number of upper-case letters = 3	As expected



Test number	Purpose	Input	Expected Result	Actual Result
16	To verify that this method doesn't count lower-case letters	“hello. how are you? i'm good thanks!”	Number of upper-case letters = 0	As expected



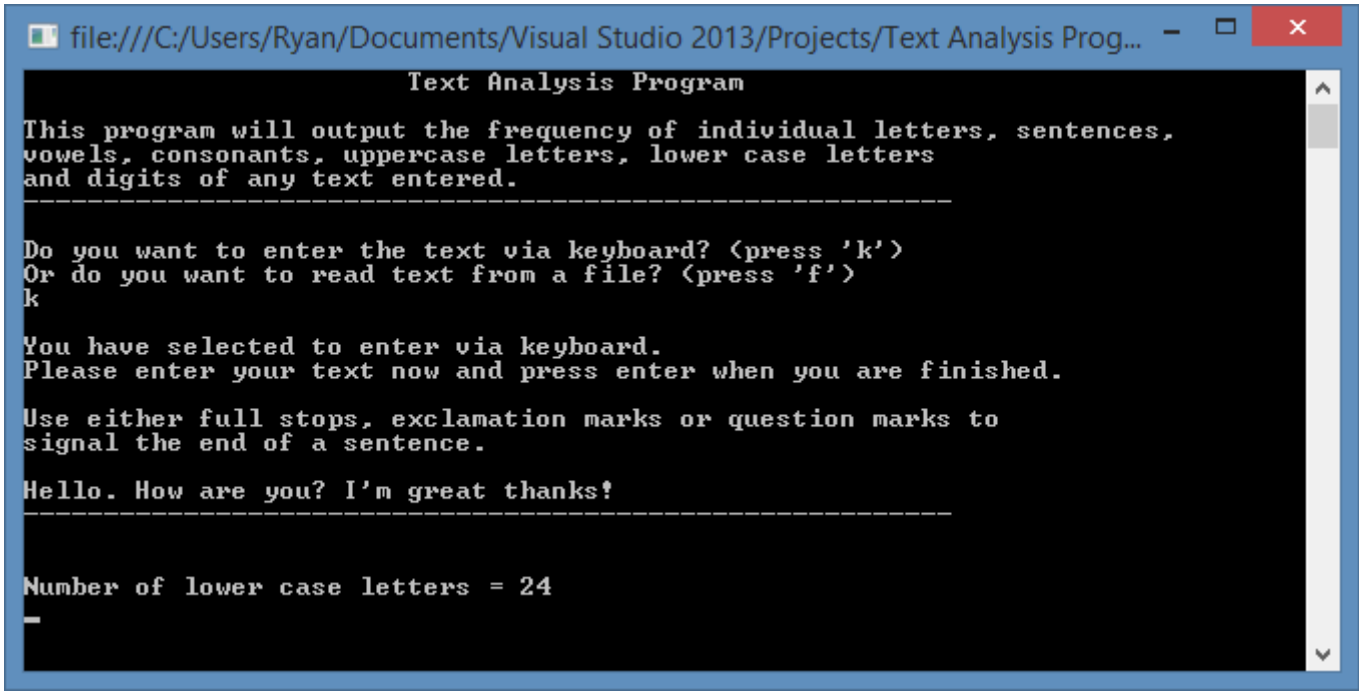
Test number	Purpose	Input	Expected Result	Actual Result
17	To verify that this method doesn't count lower-case letters or other characters	“abcdefghijklmnopqrstuvwxyz ##£$&”	Number of upper-case letters = 0	As expected



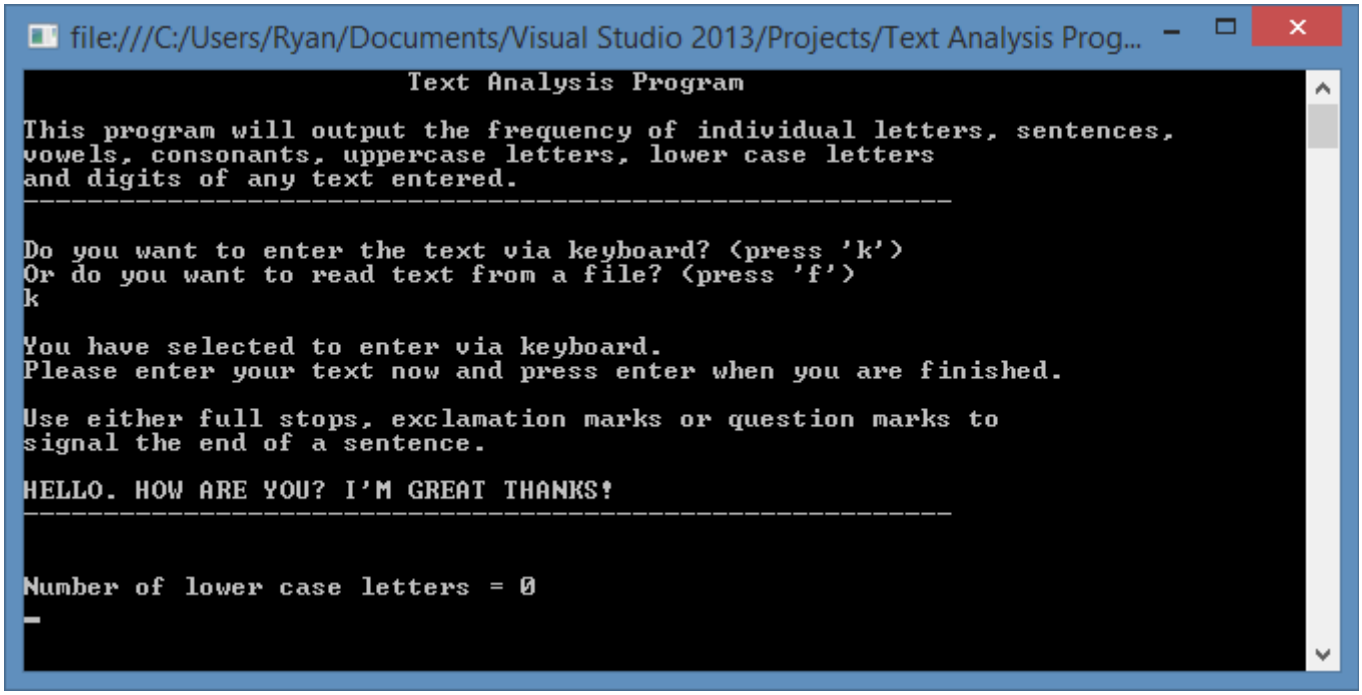
Black Box Testing – NumOfLowerCase()

For this test, I will enter a normal/expected sentence with a mix of upper-case and lower-case letters, then just lower-case letters and then finally just upper-case/other invalid characters.

Test number	Purpose	Input	Expected Result	Actual Result
18	To verify that this method only counts lower-case letters correctly in a normal series of sentences	“Hello. How are you? I'm great thanks!”	Number of lower-case letters = 24	As expected



Test number	Purpose	Input	Expected Result	Actual Result
19	To verify that this method doesn't count upper-case letters	"HELLO. HOW ARE YOU? I'M GREAT THANKS!"	Number of lower-case letters = 0	As expected

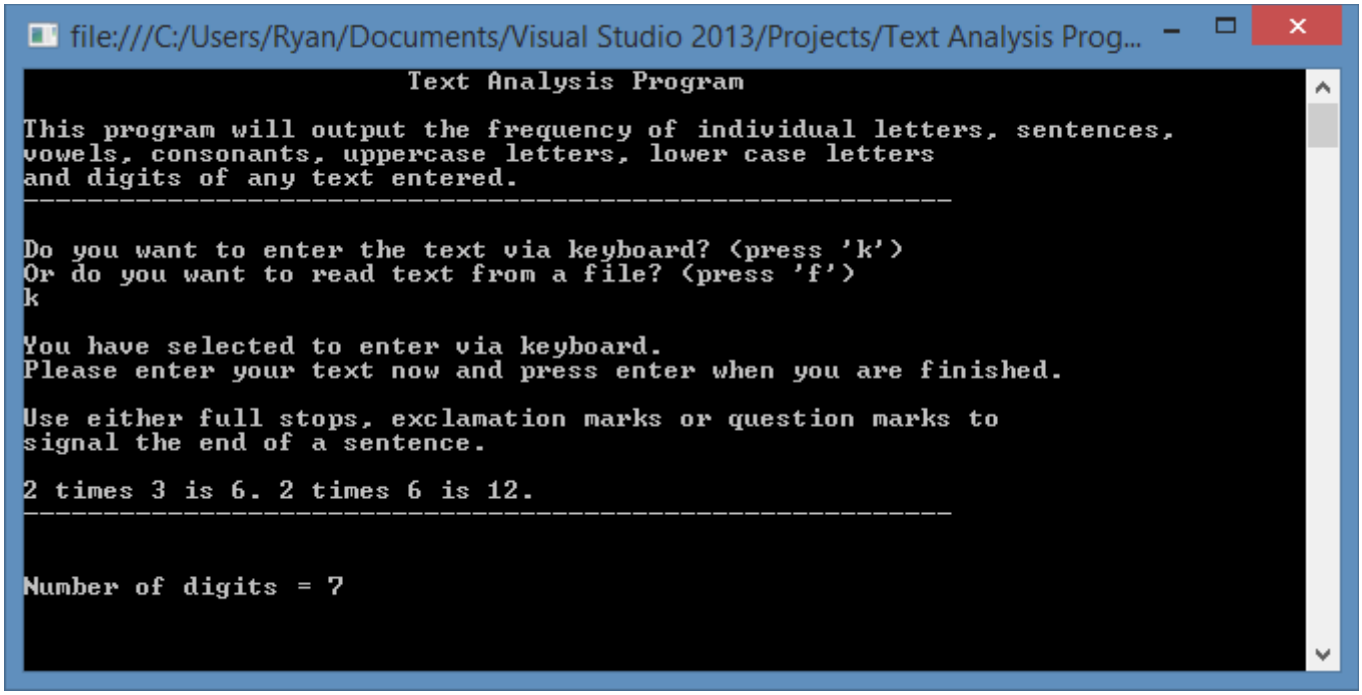


Test number	Purpose	Input	Expected Result	Actual Result
20	To verify that this method doesn't count upper-case letters or other characters	“ABCDEFGHIJ KLMNOPQRST UVWXYZ ##£$ %&”	Number of lower-case letters = 0	As expected

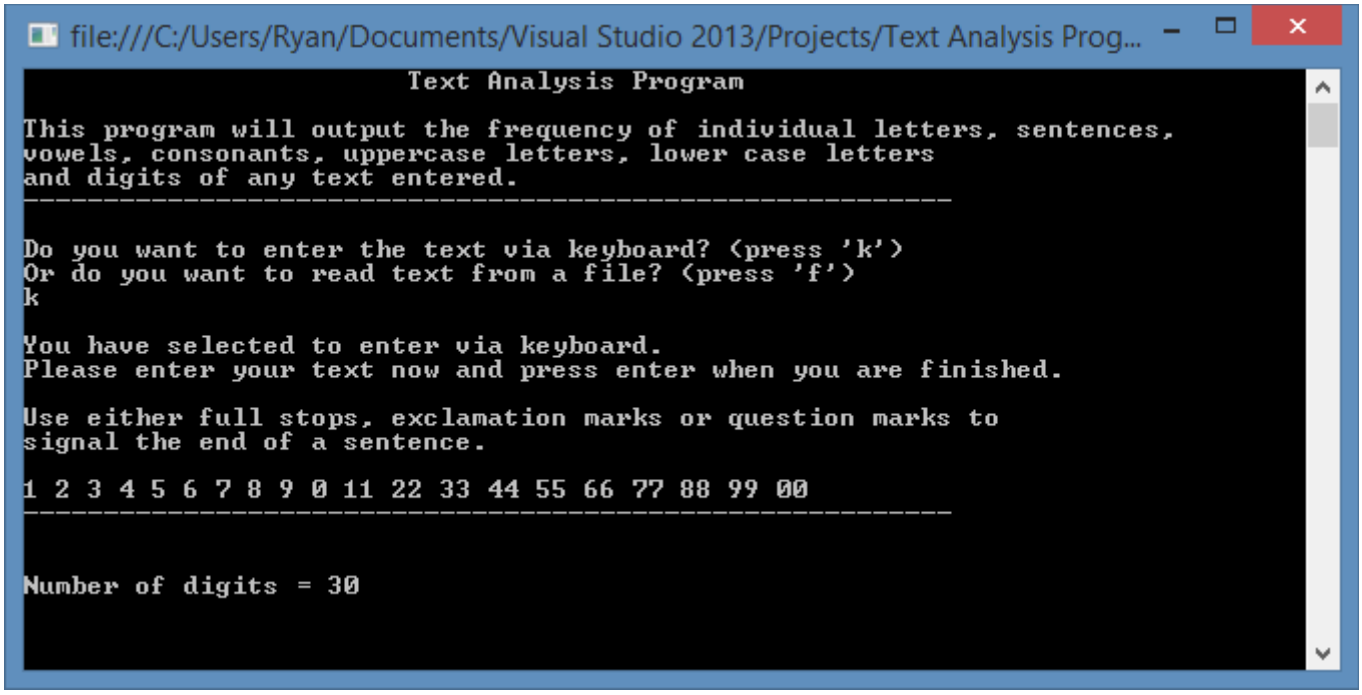
Black Box Testing – NumOfDigits()

For this test, I will enter a normal sentence with a mix of characters (including digits), then I will enter just digits and then finally I will enter a mix of characters with no digits.

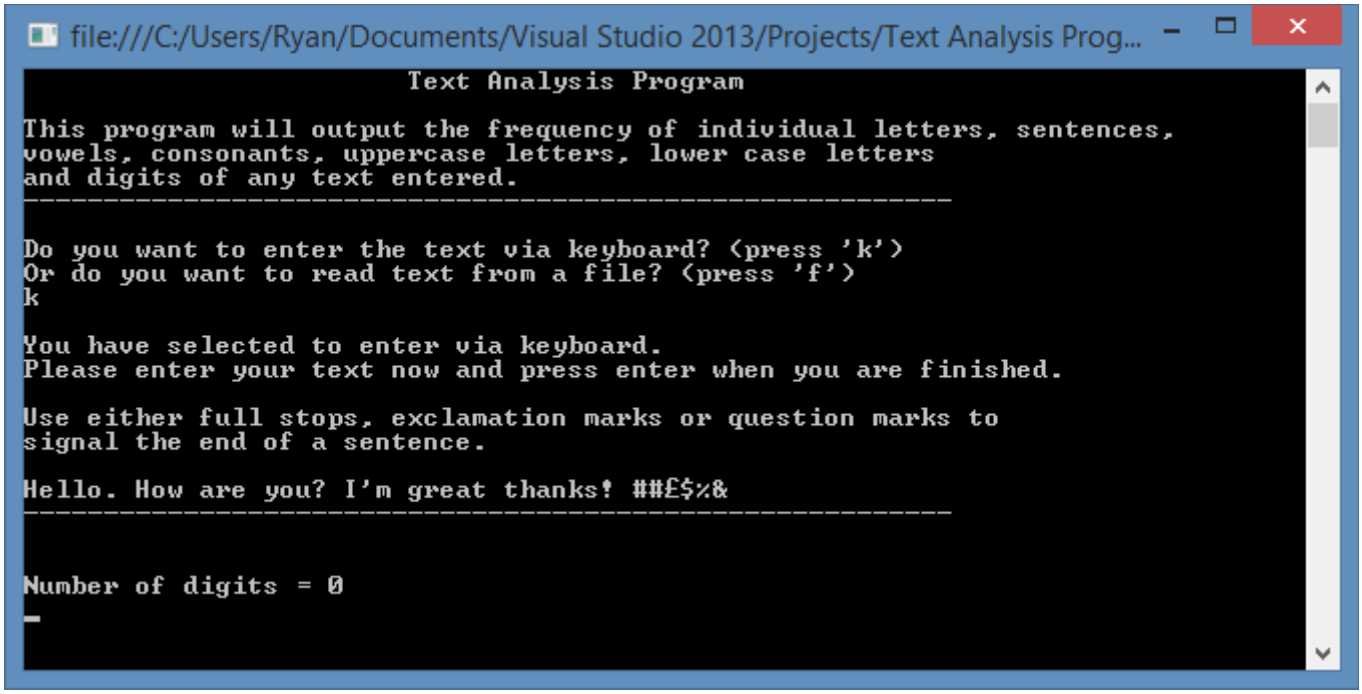
Test number	Purpose	Input	Expected Result	Actual Result
21	To verify that this method counts digits in a series of normal sentences	“2 times 3 is 6. 2 times 6 is 12”	Number of digits = 7	As expected



Test number	Purpose	Input	Expected Result	Actual Result
22	To verify that this method counts every individual digit only	“1 2 3 4 5 6 7 8 9 0 11 22 33 44 55 66 77 88 99 00”	Number of digits = 30	As expected



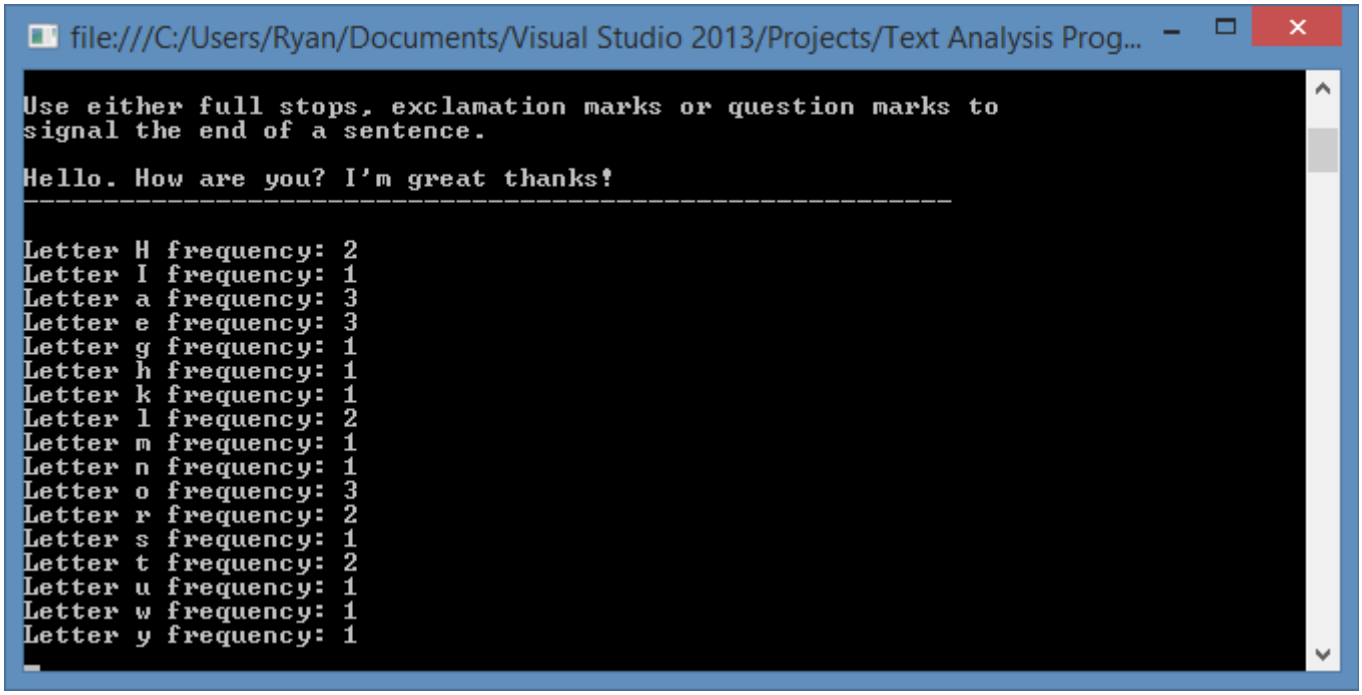
Test number	Purpose	Input	Expected Result	Actual Result
23	To verify that this method doesn't count any other characters	“Hello. How are you? I'm great thanks! ##£$ %&”	Number of digits = 0	As expected



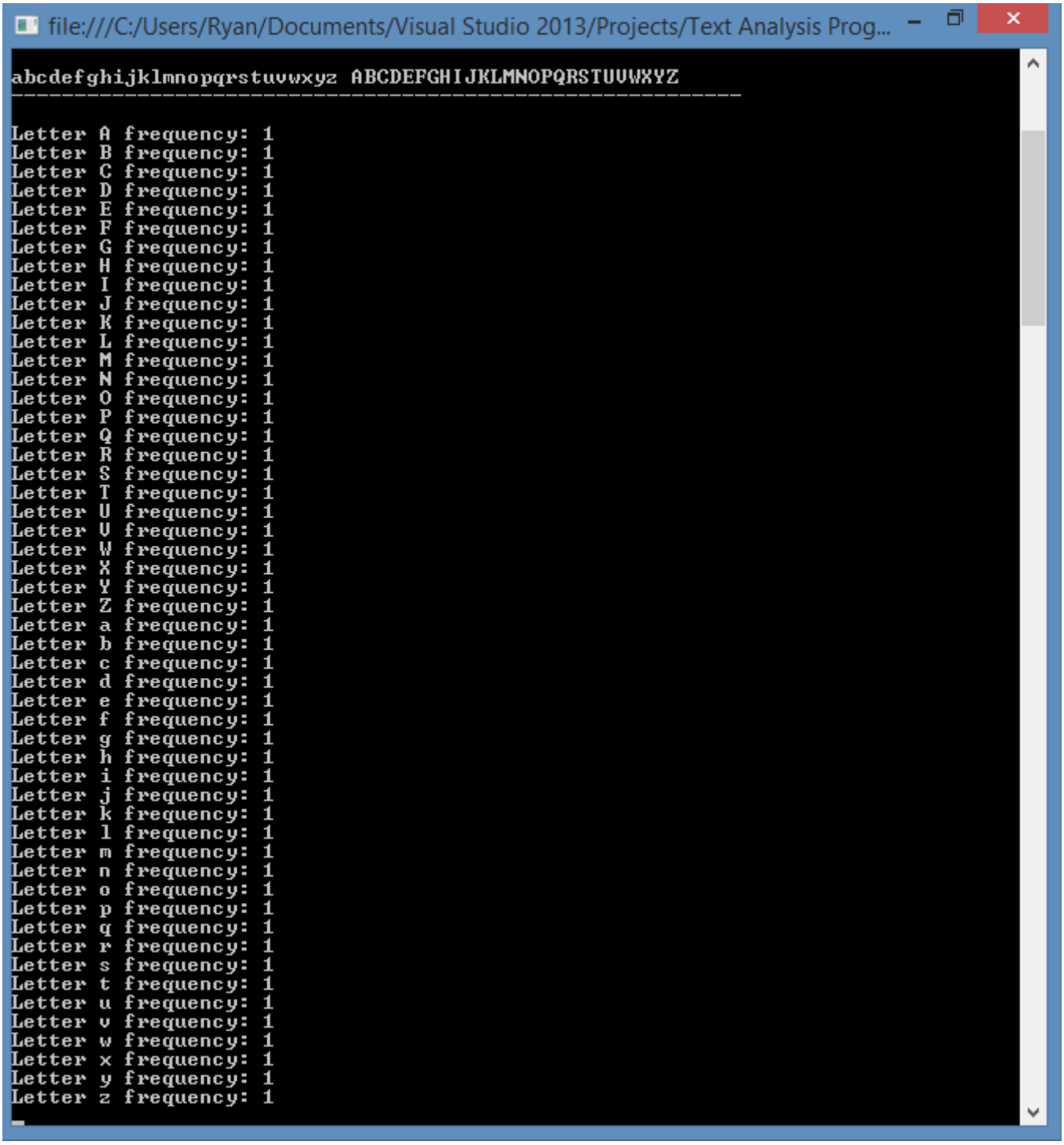
Black Box Testing – LetterFrequency()

For this test, I will enter normal sentence with a mix of different letters, then I will enter the whole alphabet and then finally I will enter a mixture of characters.

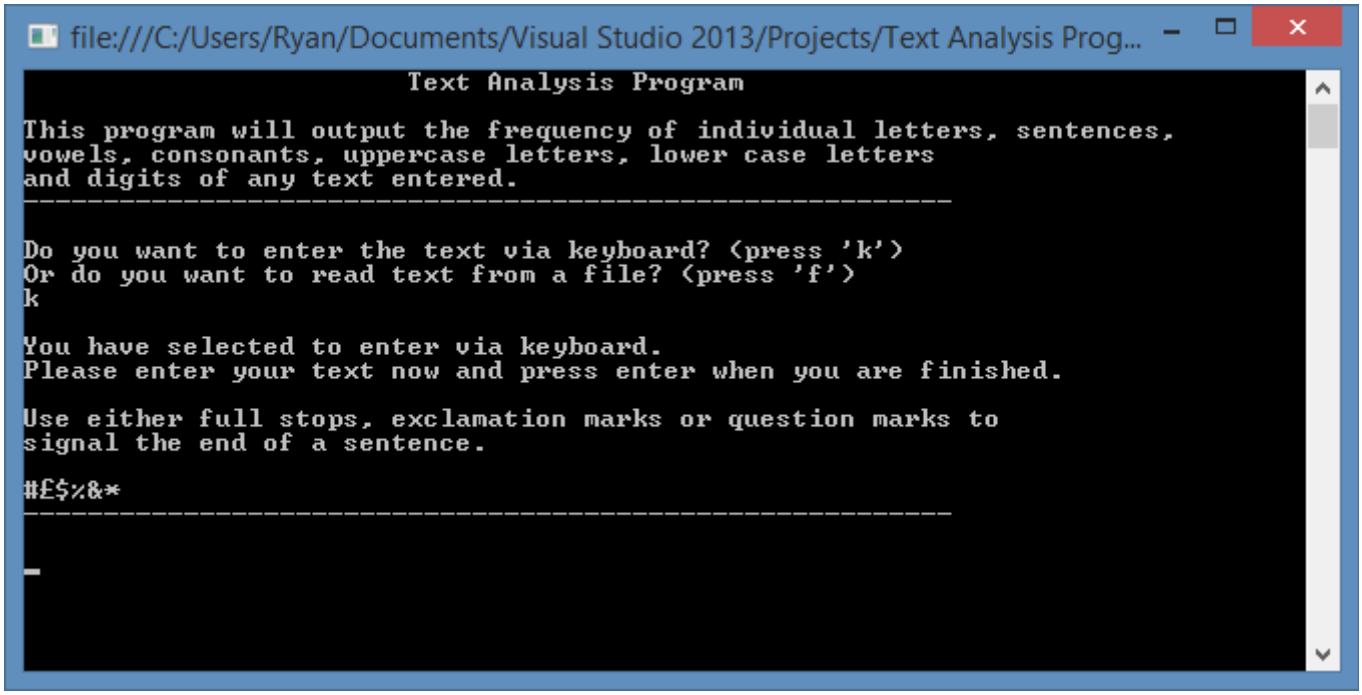
Test number	Purpose	Input	Expected Result	Actual Result
24	To verify that this method only counts letters in a series of normal sentences.	“Hello. How are you? I'm great thanks! ”	Correct frequency of each individual letter.	As expected



Test number	Purpose	Input	Expected Result	Actual Result
25	To verify that this method counts every letter of the alphabet	“abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”	Correct frequency of each individual letter.	As expected



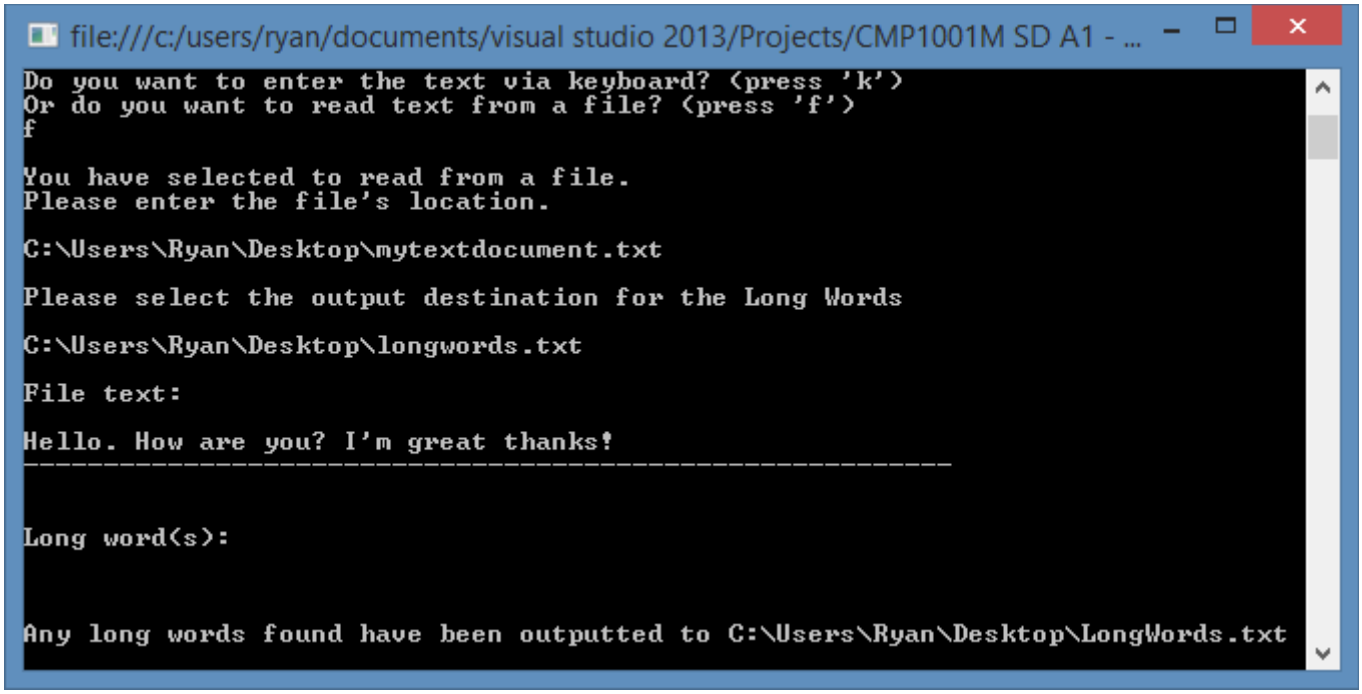
Test number	Purpose	Input	Expected Result	Actual Result
26	To verify that this method doesn't count any other characters	“???!!!£££$$$%%%&&&”	No display	As expected



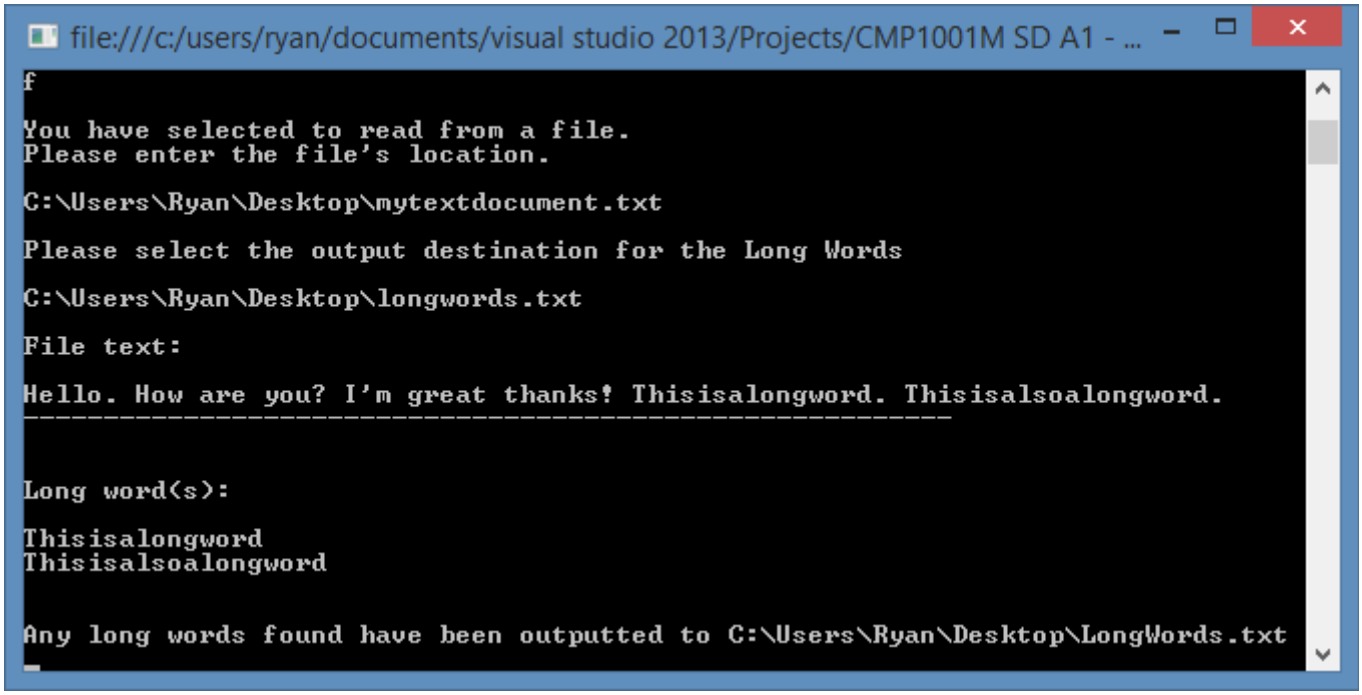
Black Box Testing – LongWords()

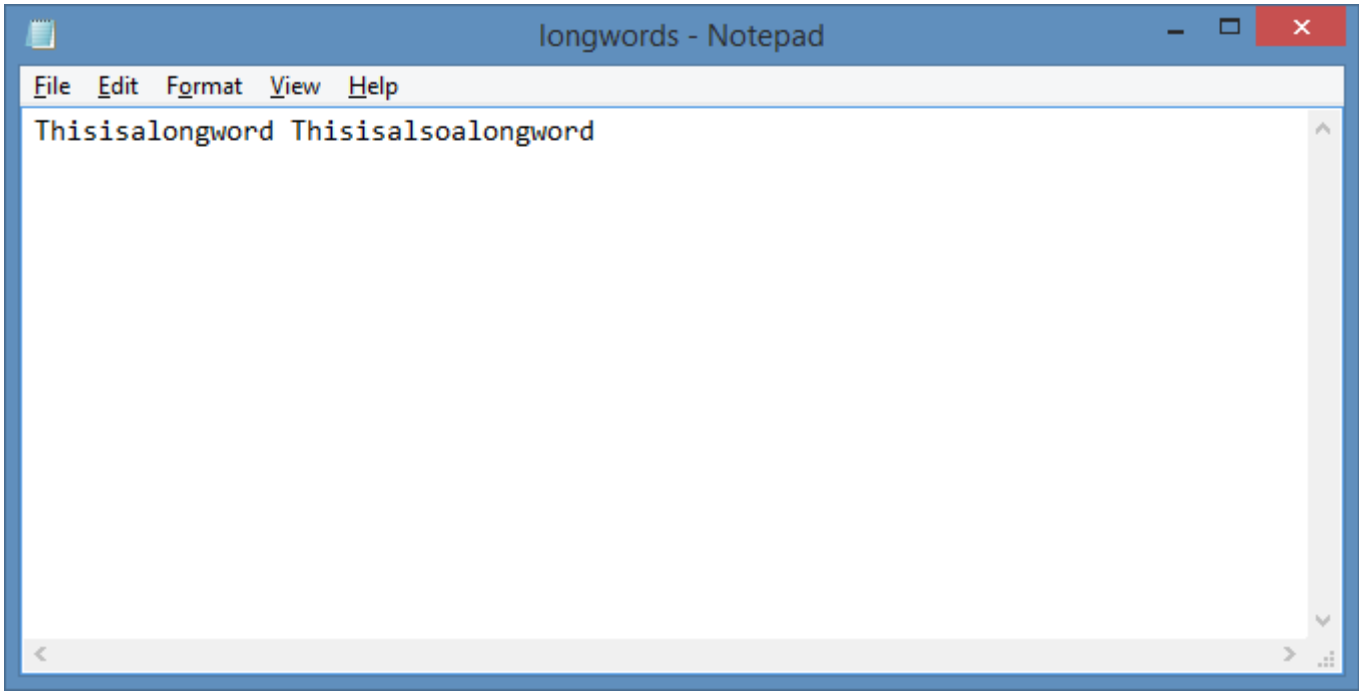
For this method, I will test it more thoroughly compared to my other methods. Firstly, I will enter a normal string with no long words (>= 7 letters), then I will enter a string with at least one long word which will include punctuation, then I will enter a string of just invalid characters.

Test number	Purpose	Input	Expected Result	Actual Result
27	To verify that this method doesn't count strings that have less than 7 letters	“”Hello. How are you? I'm great thanks!”	No display after “Long Word(s):” No outputted words	As expected

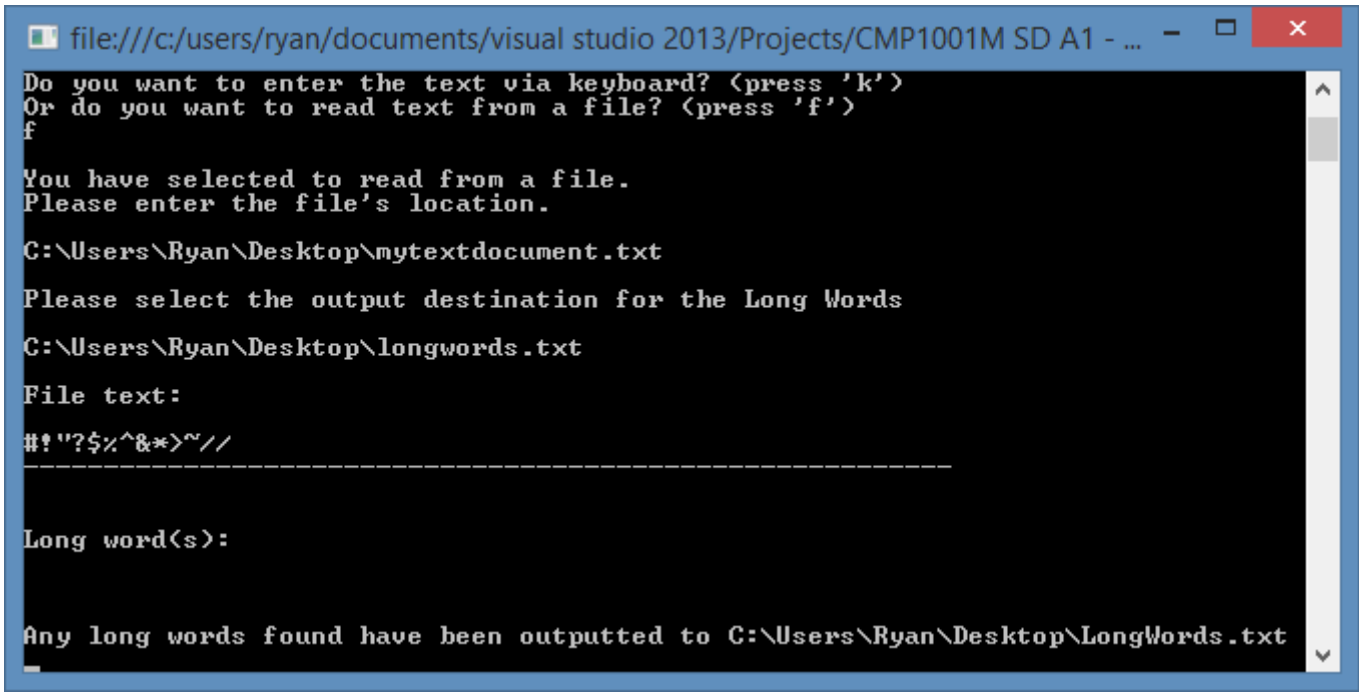


Test number	Purpose	Input	Expected Result	Actual Result
28	To verify that this method doesn't count strings that have less than 7 letters	"Hello. How are you? I'm great thanks! Thisisalongword. Thisisalsoalongword. "	Two outputs after "Long Word(s):" Two outputted words.	As expected





Test number	Purpose	Input	Expected Result	Actual Result
29	To verify that this method doesn't count invalid characters	“#!”£$%^&*//”	No display after “Long word(s):” No outputted words	As expected



White Box Testing

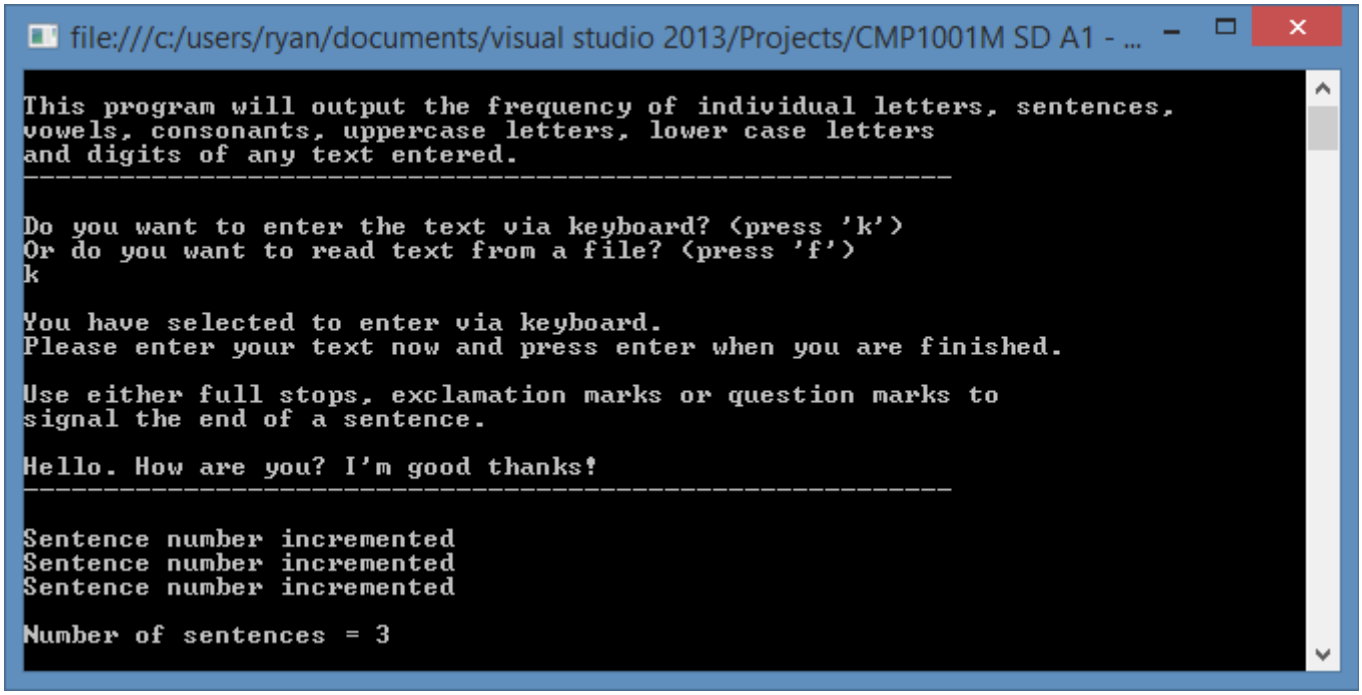
In this section, I will add temporary Console.WriteLine statements to each method individually, and then test them individually. The statements will verify that the program has entered the correct parts of the method, which will usually be if statements. If the Console.WriteLine statements aren't executed, I know that the if statement was skipped and therefore it was false.

White Box Testing – NumberOfSentences()

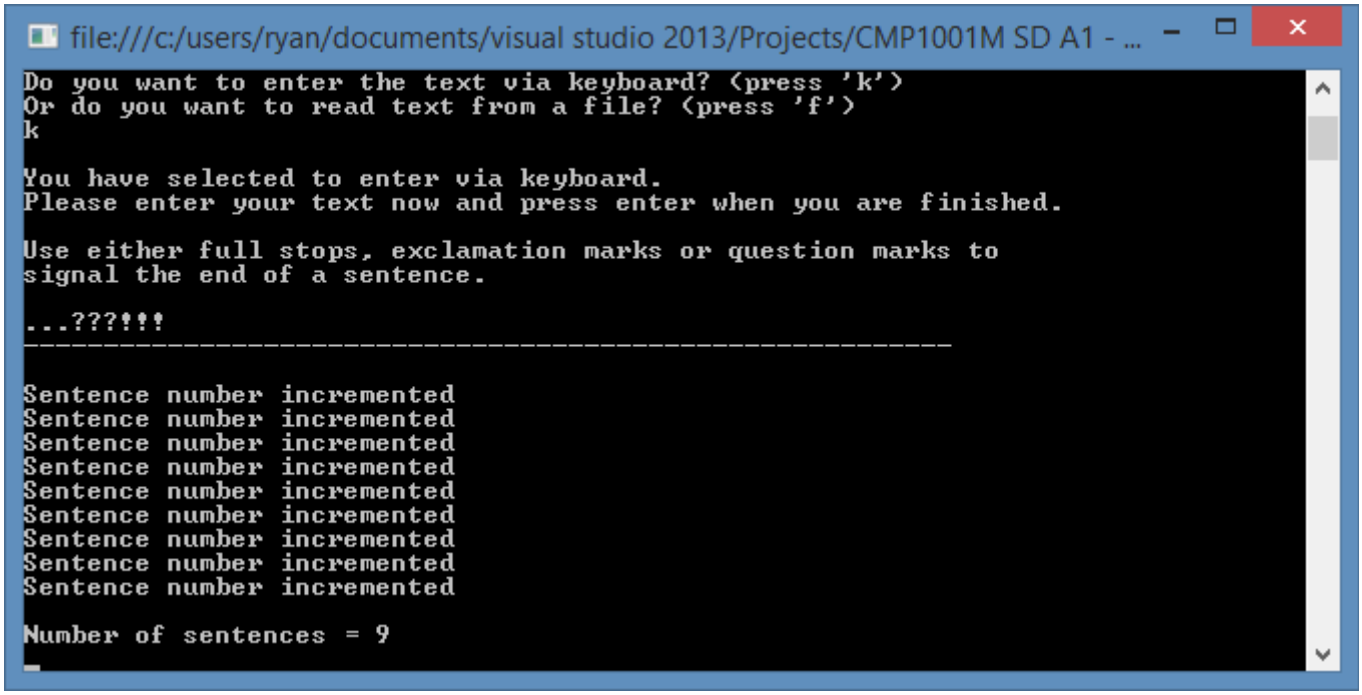
For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_semtences' gets incremented.

```
//method to calculate the number of sentences
2 references
public static int NumOfSentences()
{
    foreach (char c in Variables.text_entered)
    {
        if (c == '.' || c == '?' || c == '!')
        {
            Console.WriteLine("Sentence number incremented");
            Variables.num_of_sentences++;
        }
    }
    return Variables.num_of_sentences;
}
```

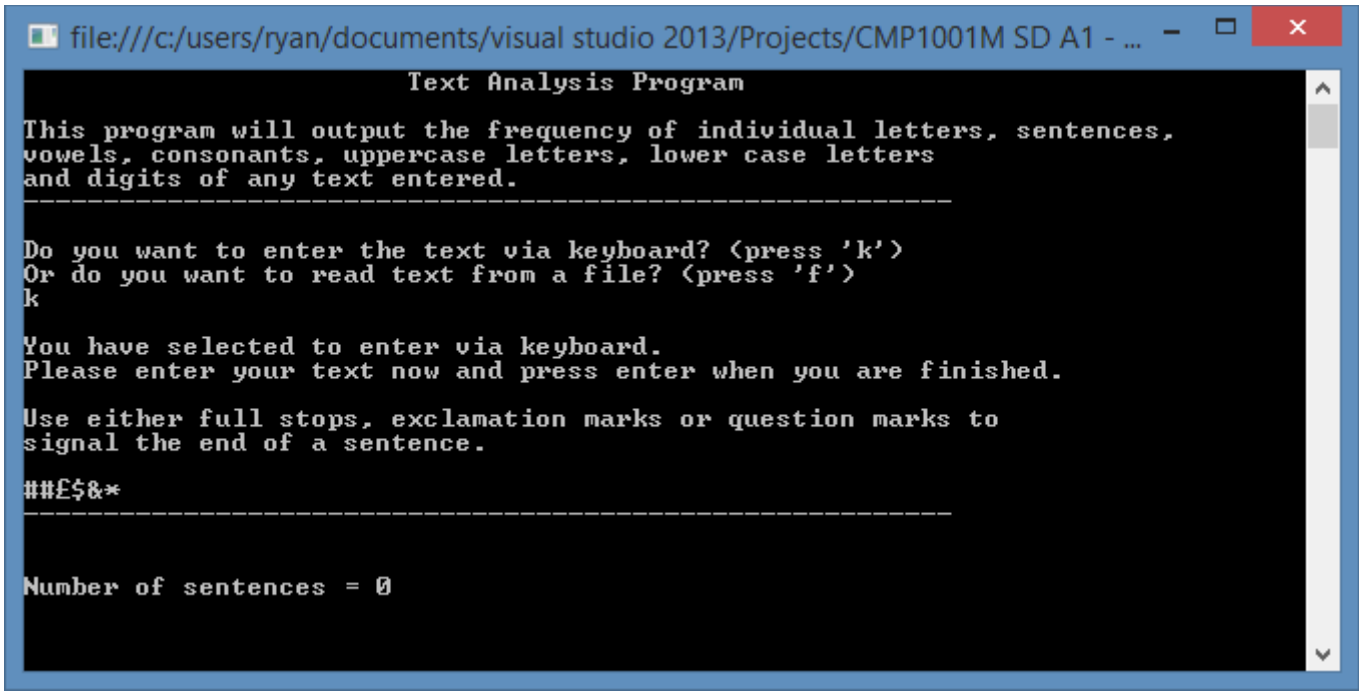
Test number	Purpose	Input	Expected Result	Actual Result
30	To verify that the correct parts of the method are used depending on the input	“Hello. How are you? I'm good thanks!”	“Sentence number incremented” to appear three times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
31	To verify that the correct parts of the method are used depending on the input	"...???!!!"	"Sentence number incremented" to appear nine times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
32	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“##£$%&*”	“Sentence number incremented” to appear zero times	As expected

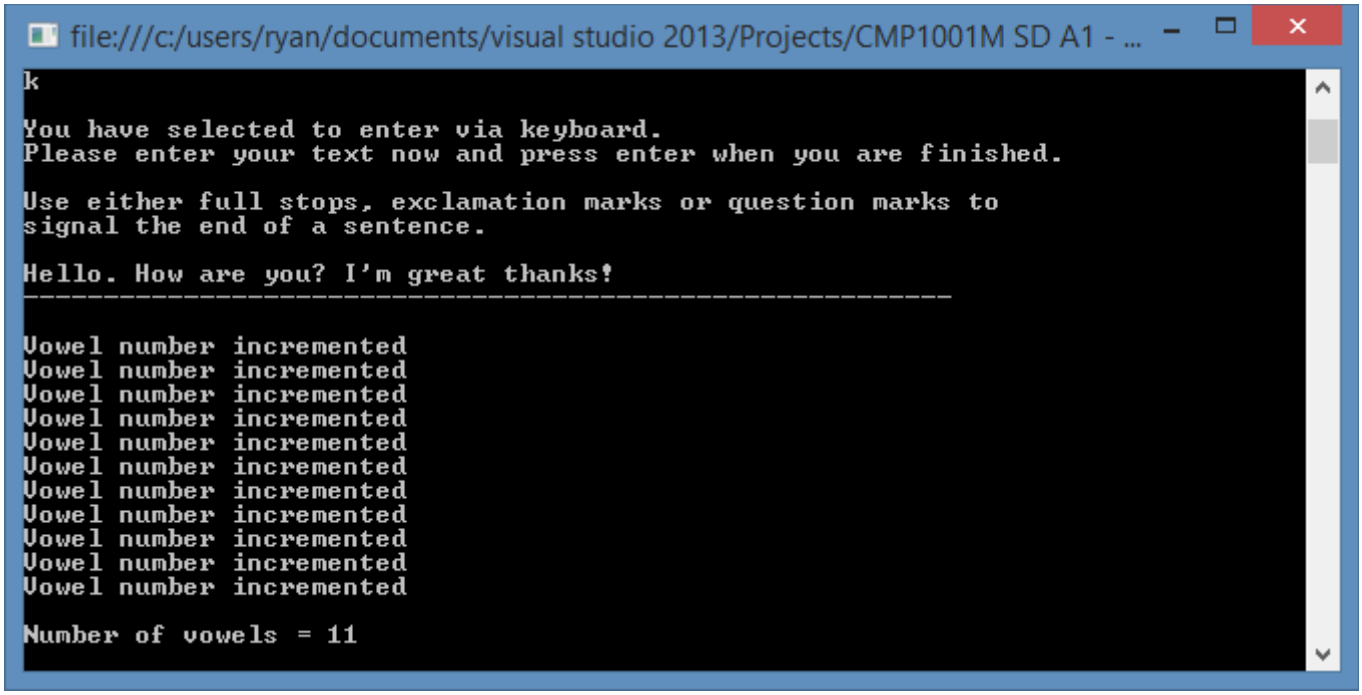


White Box Testing – NumOfVowels()

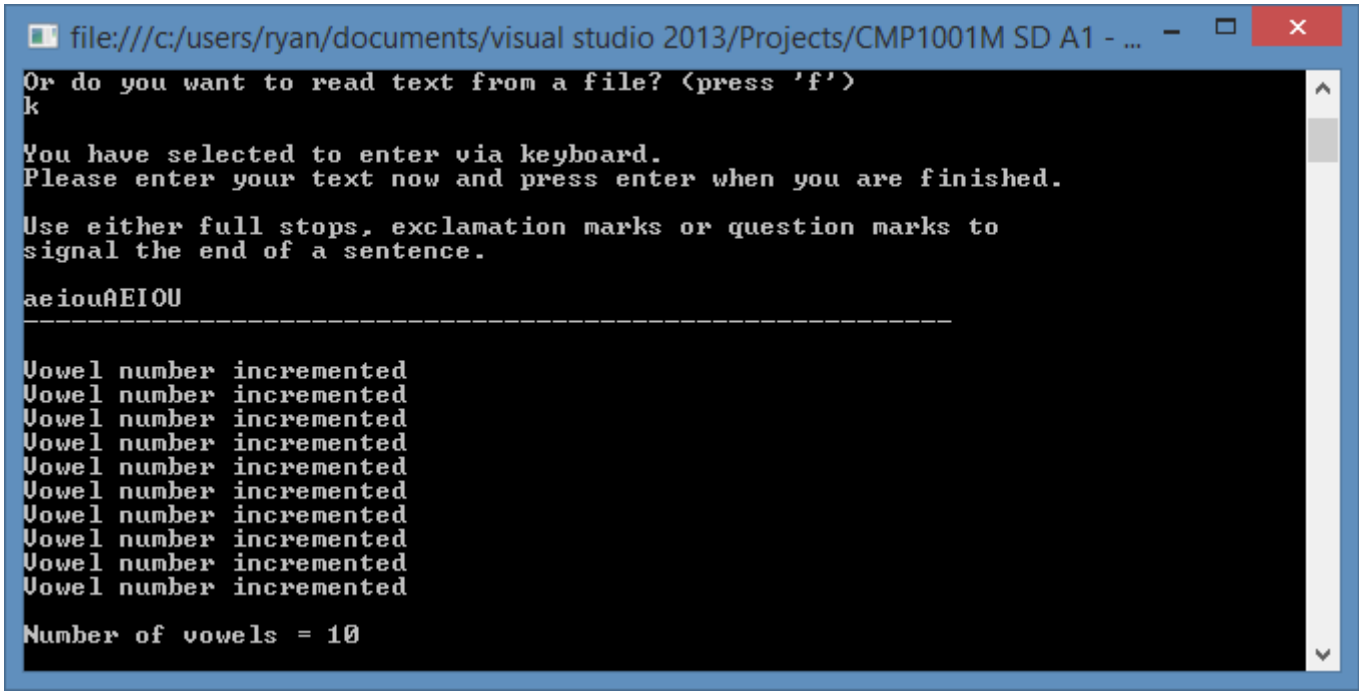
For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_vowels' gets incremented.

```
foreach (char v in Variables.text_entered)
{


    if (v == 'a' || v == 'A' || v == 'e' || v == 'E' || v == 'i' || v == 'I'
        || v == 'o' || v == 'O' || v == 'u' || v == 'U')
    {
        Console.WriteLine("Vowel number incremented");
        Variables.num_of_vowels++;
    }
}
return Variables.num_of_vowels;
```

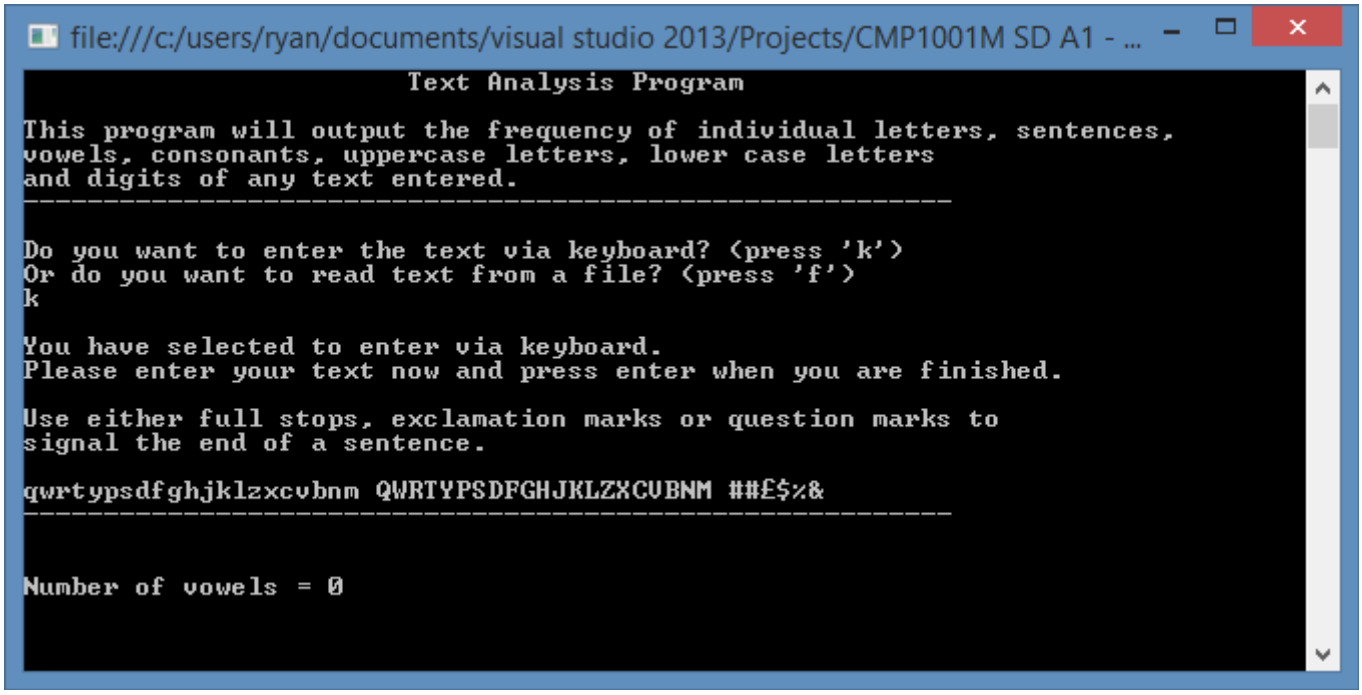
Test number	Purpose	Input	Expected Result	Actual Result
33	To verify that the correct parts of the method are used depending on the input	“Hello. How are you? I'm fine thanks!”	“Vowel number incremented” to appear eleven times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
33	To verify that the correct parts of the method are used depending on the input	“aeiouAEIOU”	“Vowel number incremented” to appear ten times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
34	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“qwrtypsdfghjlzxcv bnm QWRTYPSDFGHJ KLZXCVBNM ##£ $%&”	“Vowel number incremented” to appear zero times	As expected

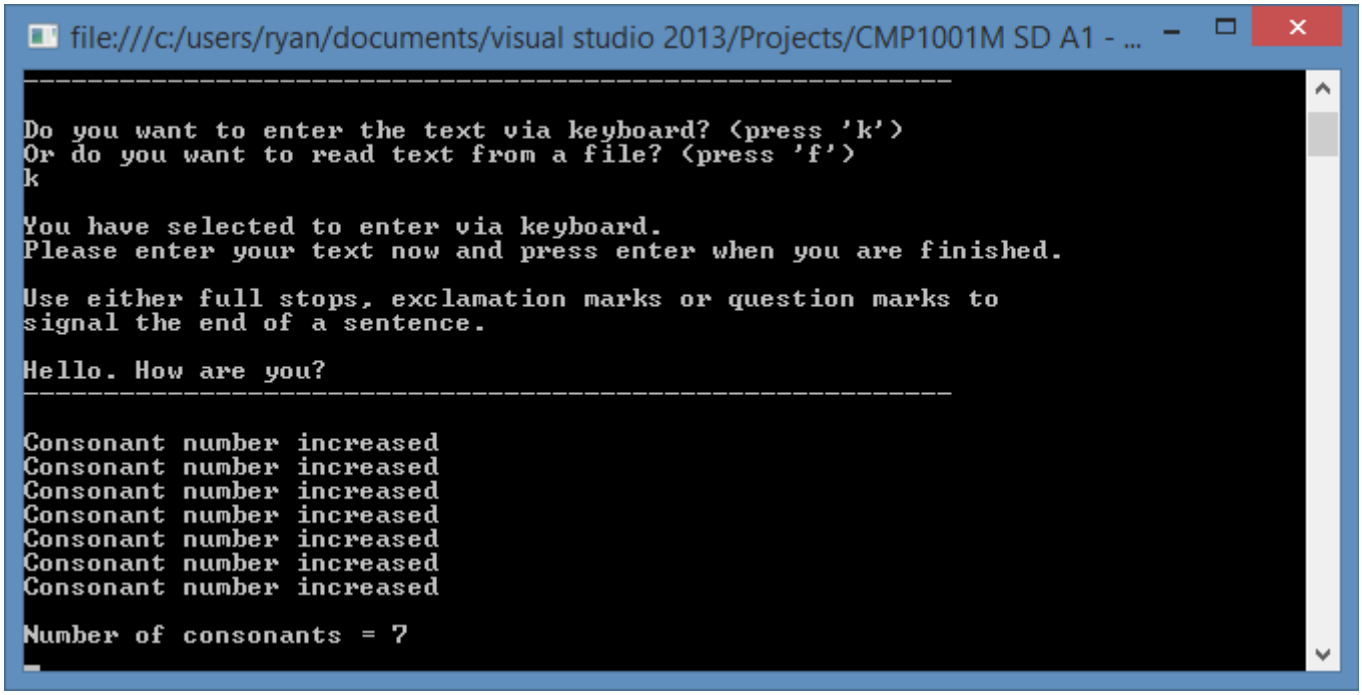


White Box Testing – NumOfConsonants()

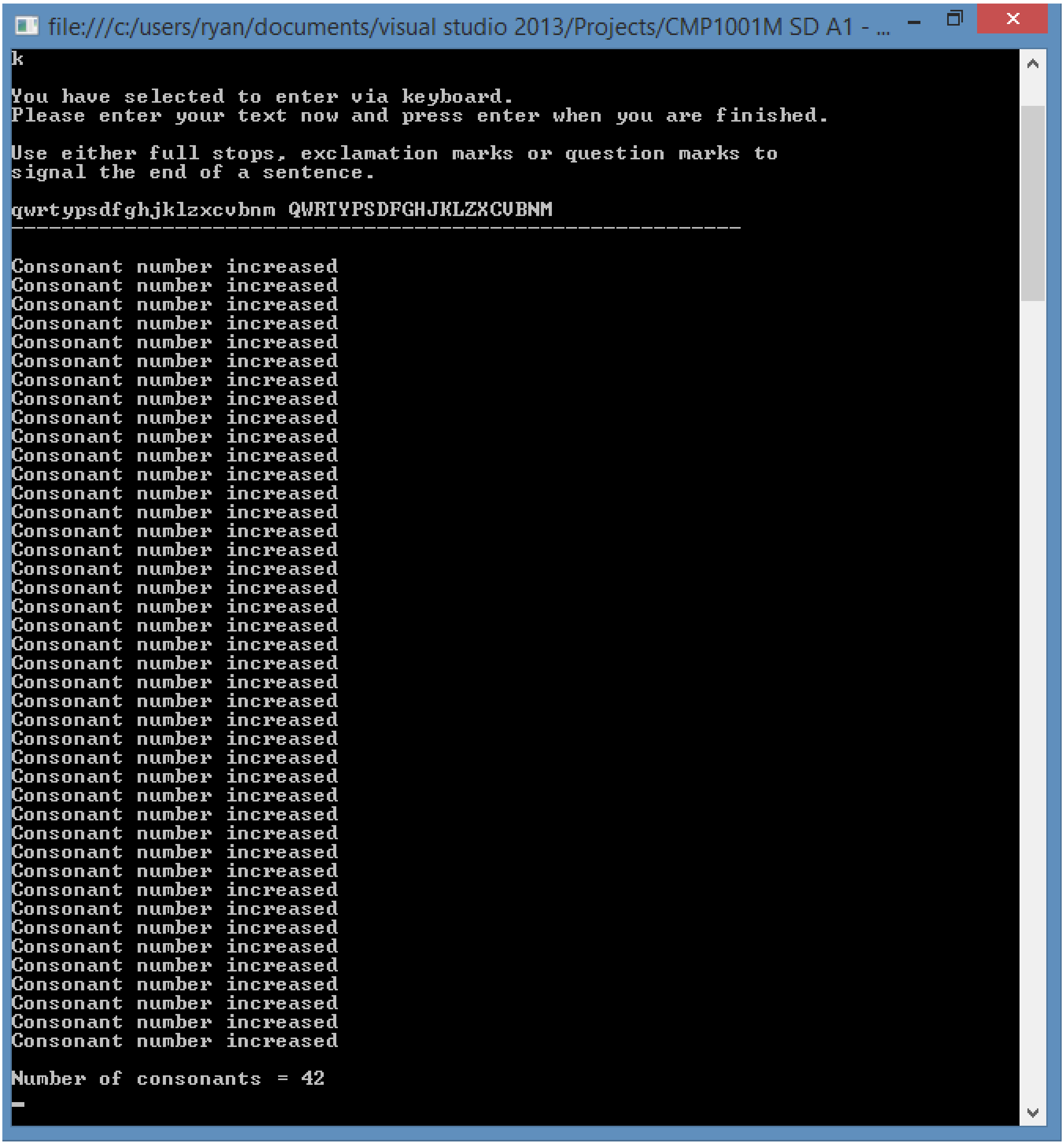
For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_consonants' gets incremented.

```
//method to count the number of consonants
2 references
public static int NumOfConsonants()
{
    foreach (char c in Variables.text_entered)
    {
        if (c == 'b' || c == 'B' || c == 'c' || c == 'C' || c == 'd' || c == 'D'
            || c == 'f' || c == 'F' || c == 'g' || c == 'G' || c == 'h' || c == 'H'
            || c == 'j' || c == 'J' || c == 'k' || c == 'K' || c == 'l' || c == 'L'
            || c == 'm' || c == 'M' || c == 'n' || c == 'N' || c == 'p' || c == 'P'
            || c == 'q' || c == 'Q' || c == 'r' || c == 'R' || c == 's' || c == 'S'
            || c == 't' || c == 'T' || c == 'v' || c == 'V' || c == 'w' || c == 'W'
            || c == 'x' || c == 'X'|| c == 'y' || c == 'Y' || c == 'z' || c == 'Z')
        {
            Console.WriteLine("Consonant number increased");
            Variables.num_of_consonants++;
        }
    }
    return Variables.num_of_consonants;
```

Test number	Purpose	Input	Expected Result	Actual Result
35	To verify that the correct parts of the method are used depending on the input	“Hello. How are you? I'm great thanks!”	“Consonant number incremented” to appear seven times	As expected

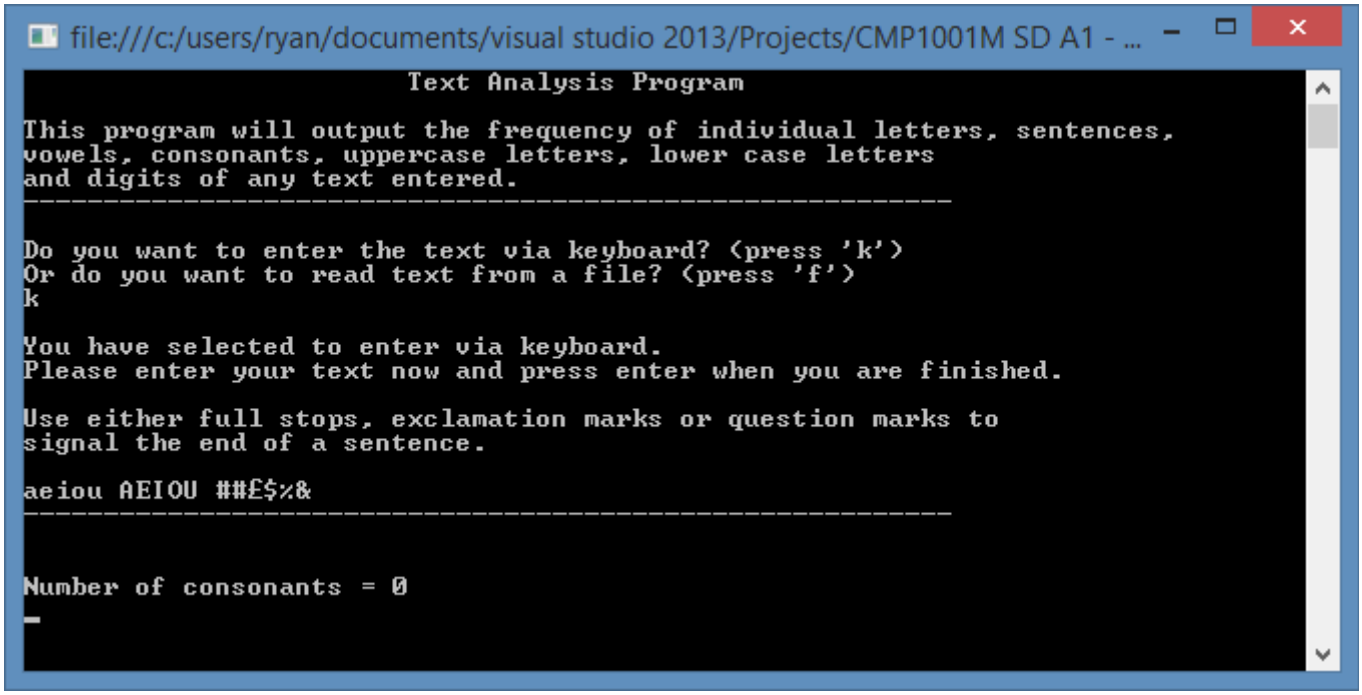


Test number	Purpose	Input	Expected Result	Actual Result
36	To verify that the correct parts of the method are used depending on the input	“qwrtypsdfghjkl QWRTYPSDFG HJKLZXCVBN M”	“Consonants number incremented” to appear forty-two times	As expected



```
file:///c:/users/ryan/documents/visual studio 2013/Projects/CMP1001M SD A1 - ...
k

You have selected to enter via keyboard.
Please enter your text now and press enter when you are finished.

Use either full stops, exclamation marks or question marks to
signal the end of a sentence.

qwrtypsdfghjklzxcvbnm QWRTYPSDFGHJKLZXCVBNM
-----------------------------------------------------------

Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased
Consonant number increased

Number of consonants = 42
_
```

Test number	Purpose	Input	Expected Result	Actual Result
37	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“aeiou AEIOU ##£ $%&”	“Consonants number incremented” to appear zero times	As expected

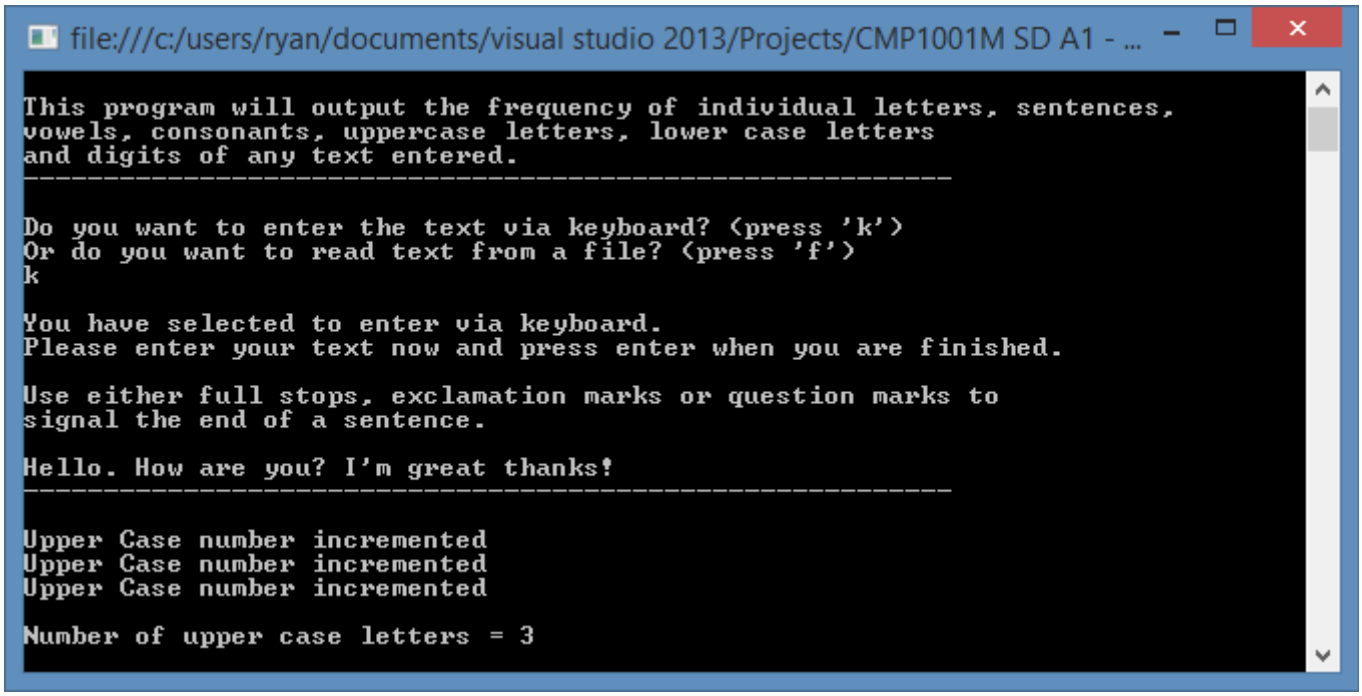


White Box Testing – NumOfUpperCase()

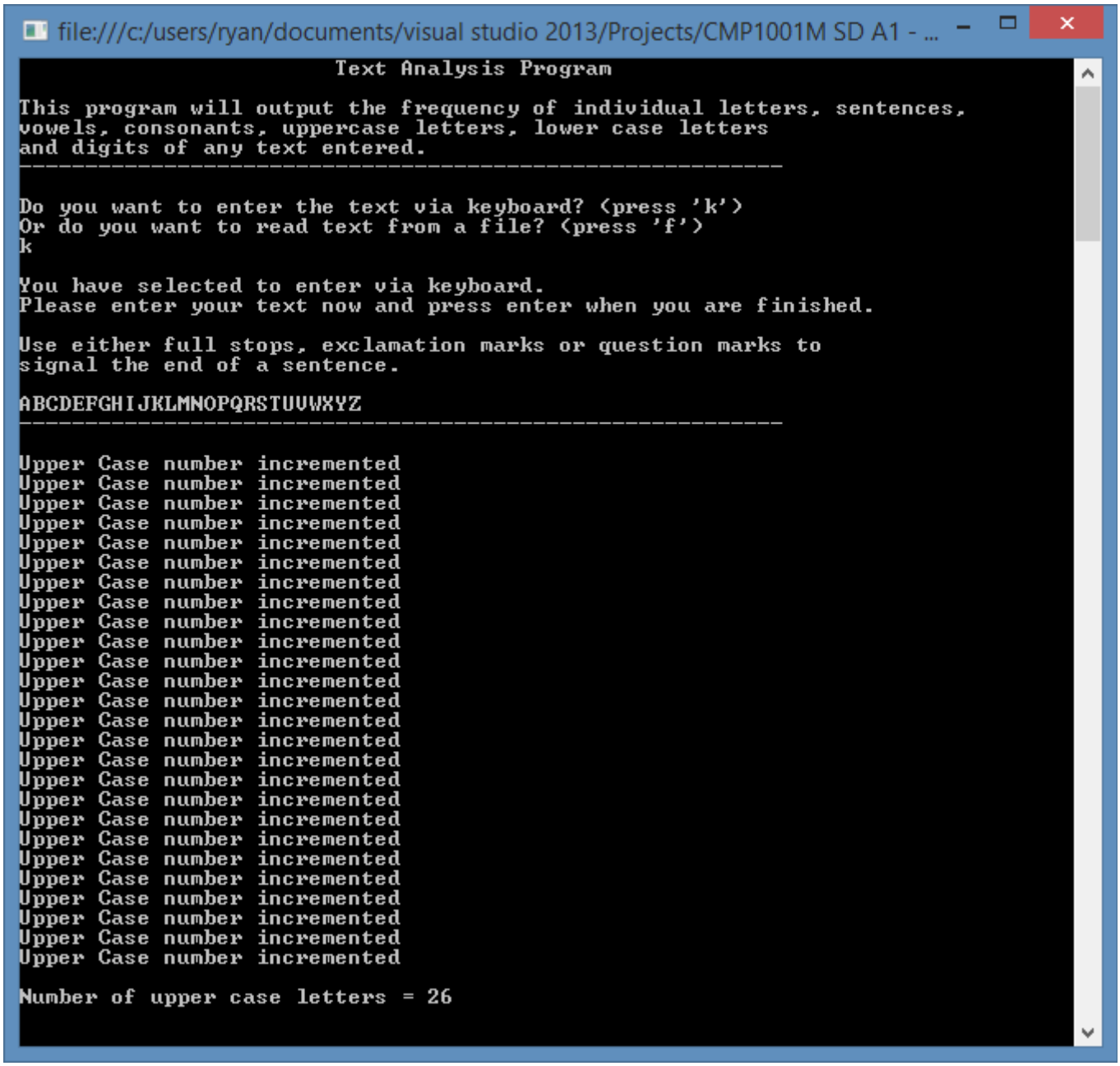
1. For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_uppercase' gets incremented.

```
//method for counting the number of uppercase letters
2 references
public static int NumOfUpperCase()
{
    for (int i = 0; i < Variables.text_entered.Length; i++)
    {
        //Counting the number of uppercase letters using the IsUpper function
        if (char.IsUpper(Variables.text_entered[i]))
        {
            Console.WriteLine("Upper Case number incremented");
            Variables.num_of_uppercase++;
        }
    }
    return Variables.num_of_uppercase;
```

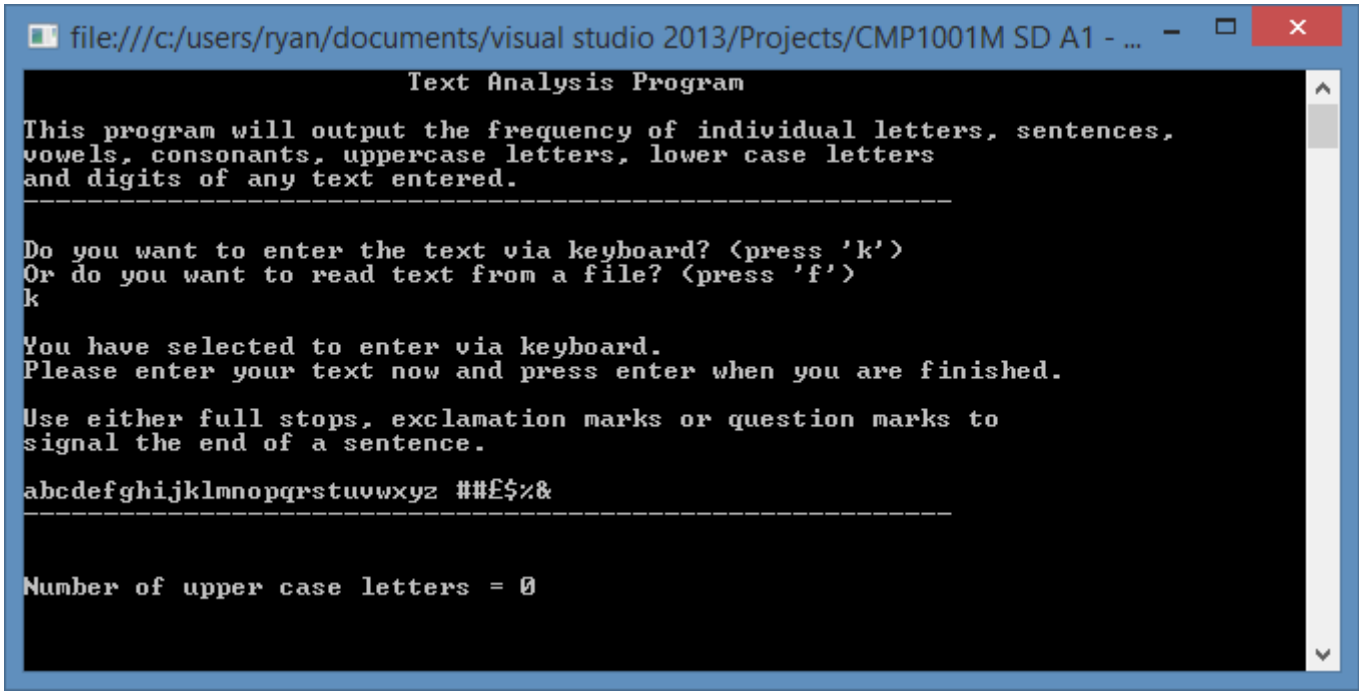
Test number	Purpose	Input	Expected Result	Actual Result
38	To verify that the correct parts of the method are used depending on the input	“Hello. How are you? I'm great thanks!”	“Upper case number incremented” to appear three times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
39	To verify that the correct parts of the method are used depending on the input	"ABCDEFGHIJKLMNOPQRSTUVWXYZ"	"Upper case number incremented" to appear twenty-six times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
40	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“abcdefghijklmnopqrstuvwxyz ##£$%&”	“Upper case number incremented” to appear zero times	As expected

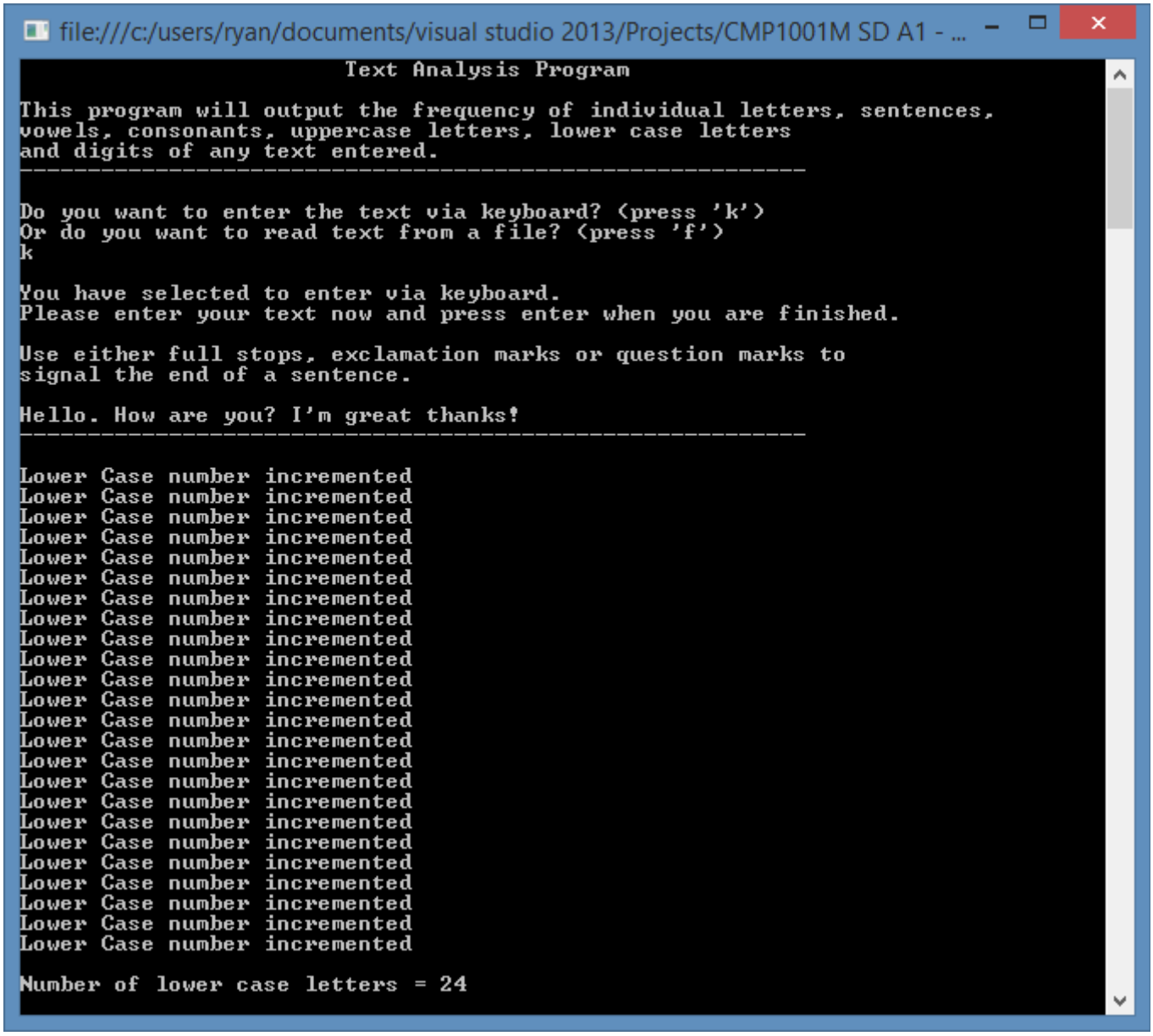


White Box Testing – NumOfLowerCase()

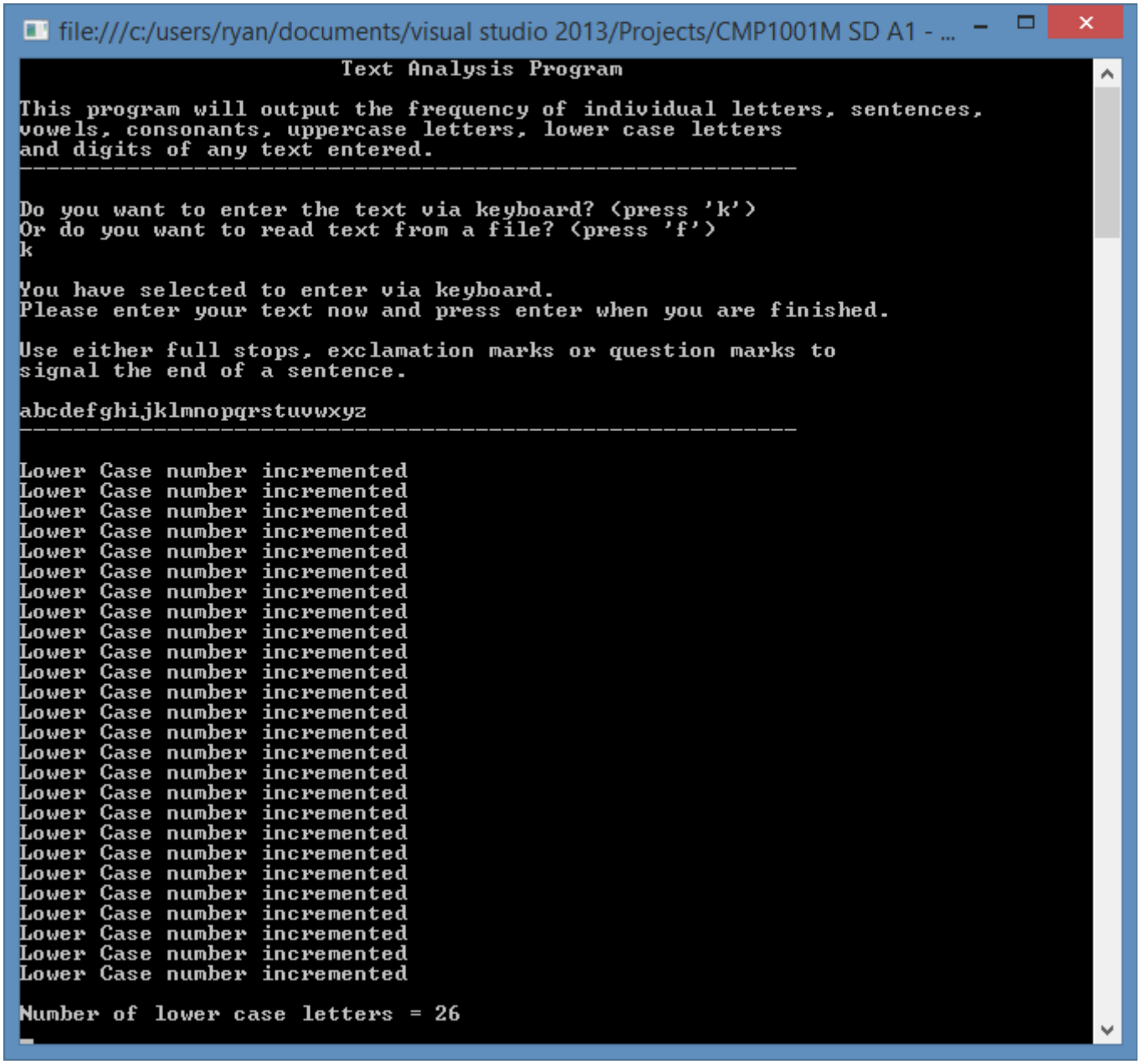
For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_lowercase' gets incremented.

```
//method for counting the number of lowercase letters
2 references
public static int NumOfLowerCase()
{
    for (int i = 0; i < Variables.text_entered.Length; i++)
    {
        //Counting the number of lowercase letters using the IsLower function
        if (char.IsLower(Variables.text_entered[i]))
        {
            Console.WriteLine("Lower Case number incremented");
            Variables.num_of_lowercase++;
        }
    }
    return Variables.num_of_lowercase;
```

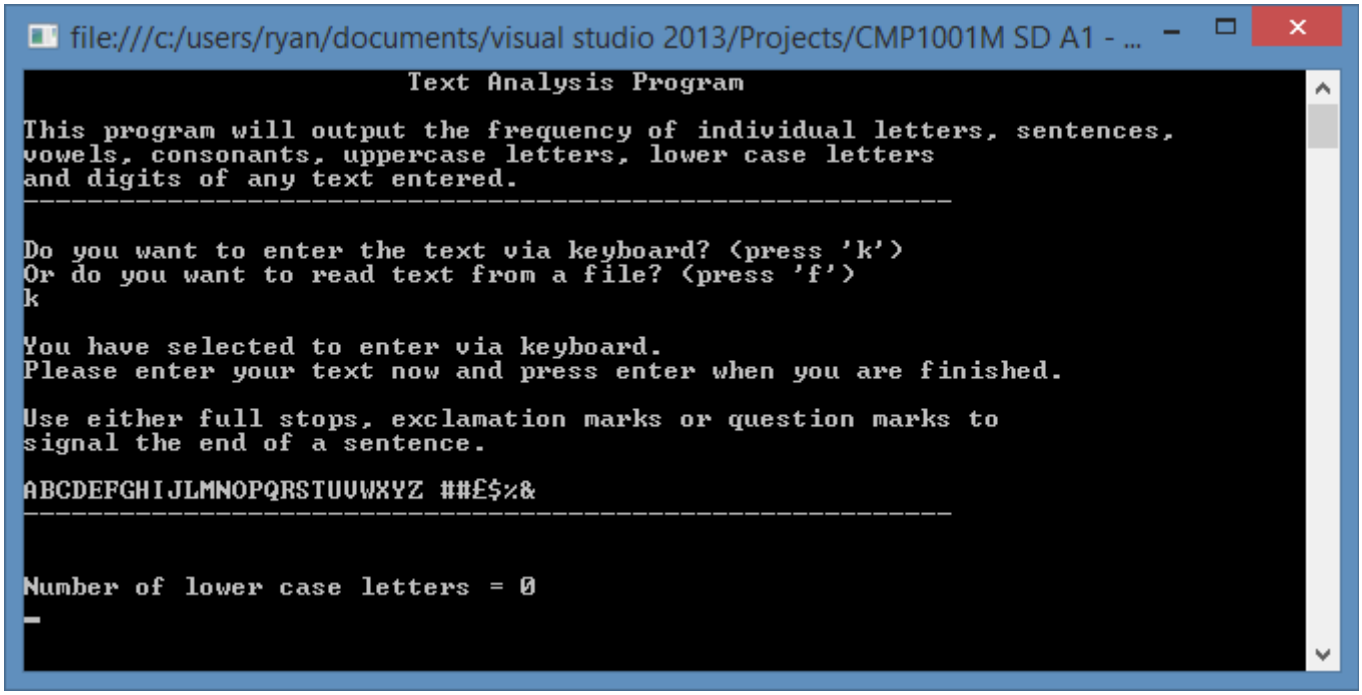
Test number	Purpose	Input	Expected Result	Actual Result
41	To verify that the correct parts of the method are used depending on the input	“Hello. How are you? I'm great thanks!”	“Lower case number incremented” to appear zero times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
42	To verify that the correct parts of the method are used depending on the input	“abcdefghijklmnopqrstuvwxyz”	“Lower case number incremented” to appear twenty-six times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
43	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“ABCDEFGHIJKLMNOP ##£$%&”	“Lower case number incremented” to appear zero times	As expected

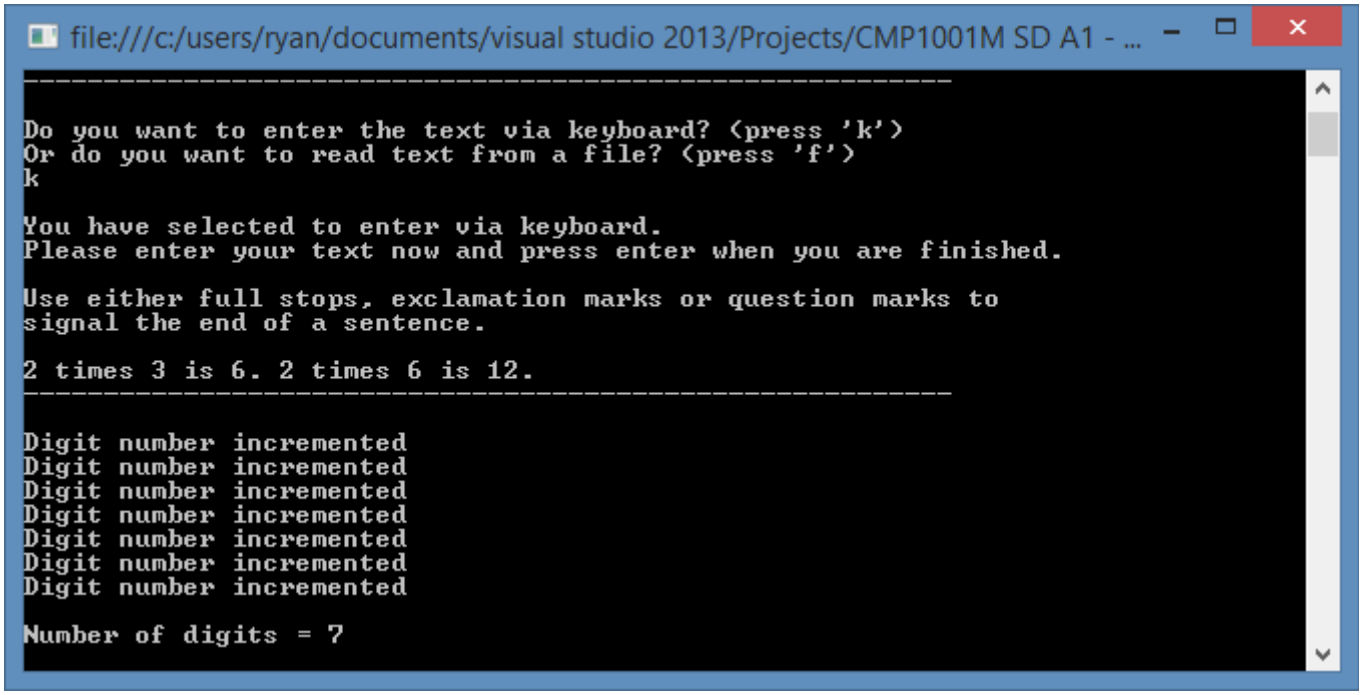


White Box Testing – NumOfDigits()

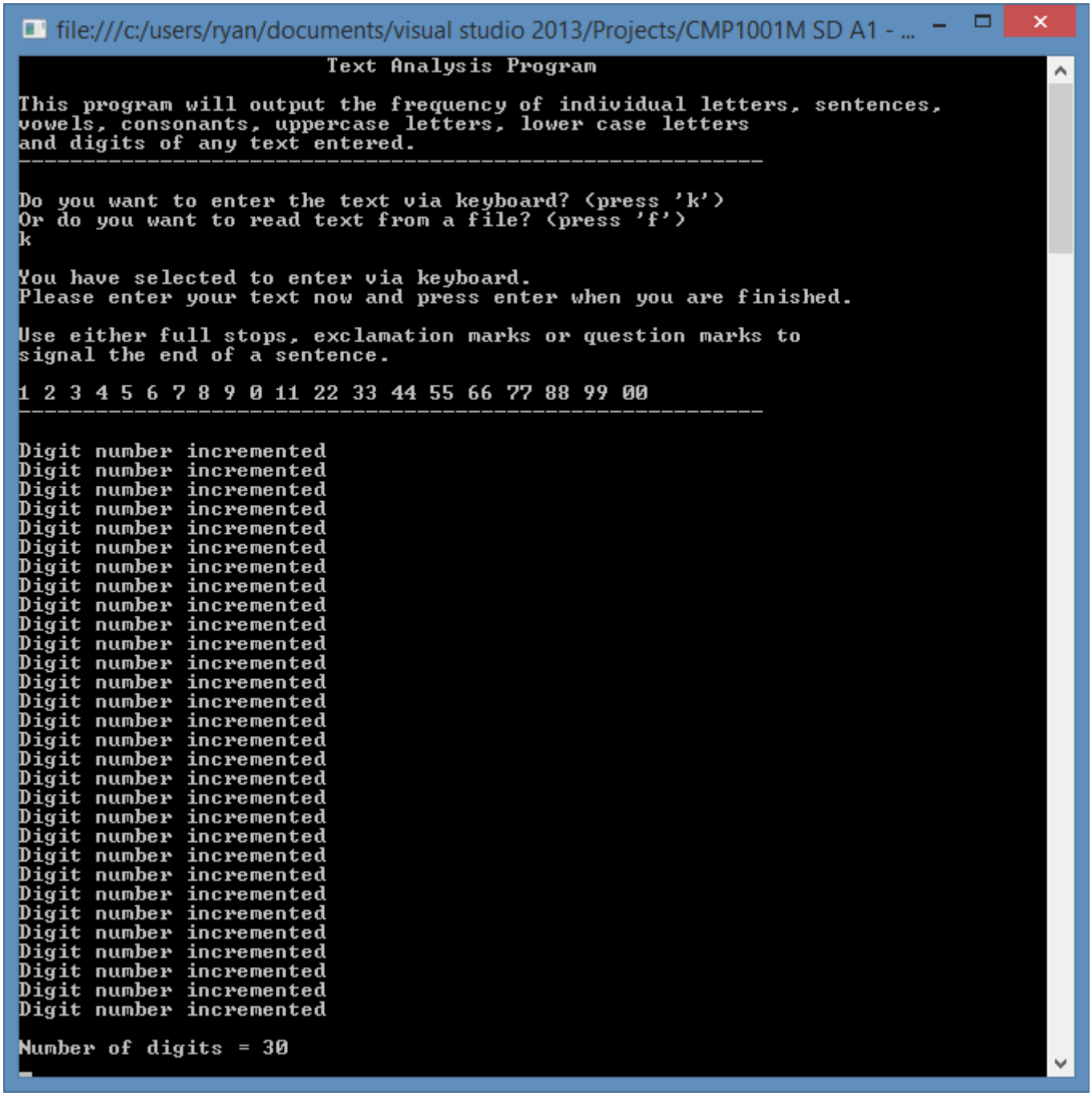
For this test, I will enter a Console.WriteLine statement above where the variable 'num_of_lowercase' gets incremented.

```
//method to cound the number of digits
2 references
public static int NumOfDigits()
{

    foreach (char d in Variables.text_entered)

        if (char.IsDigit(d))
        {
            Console.WriteLine("Digit number incremented");
            Variables.num_of_digits++;
        }
    return Variables.num_of_digits;
```

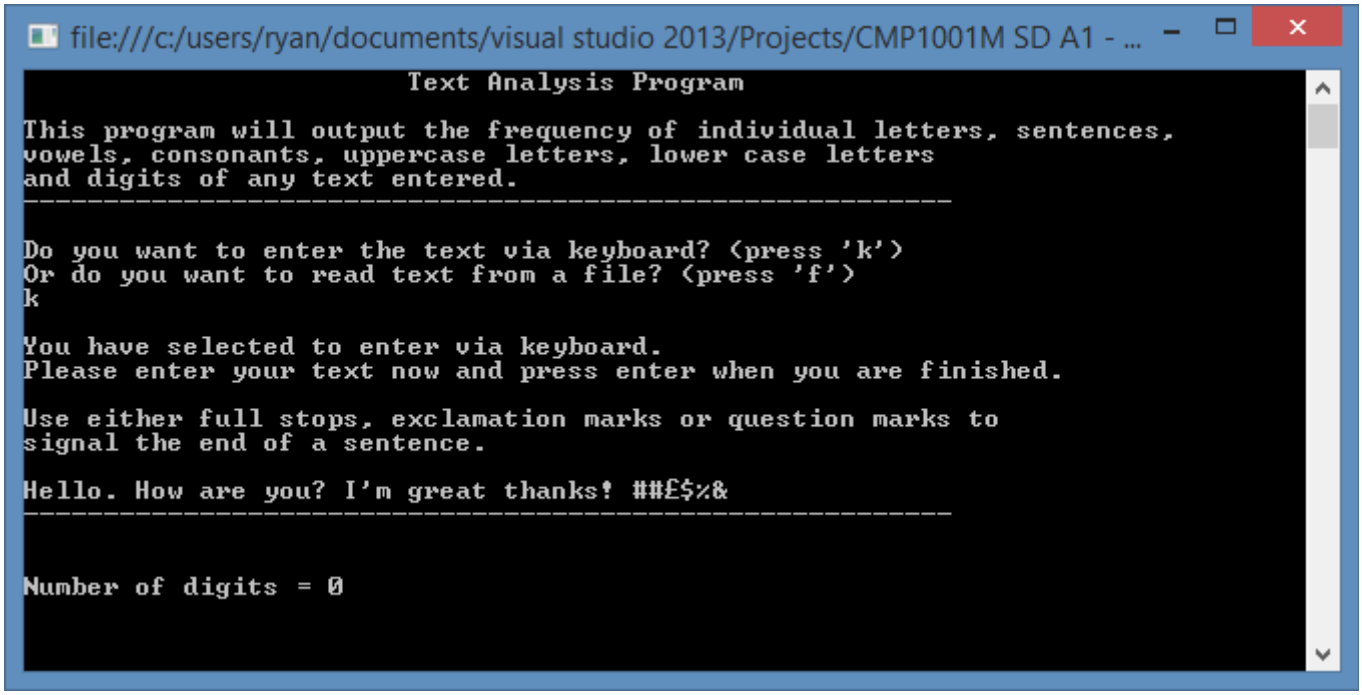
Test number	Purpose	Input	Expected Result	Actual Result
44	To verify that the correct parts of the method are used depending on the input	"2 times 3 is 6. 2 times 6 is 12."	"Digit number incremented" to appear seven times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
45	To verify that the correct parts of the method are used depending on the input	“1 2 3 4 5 6 7 8 9 0 11 22 33 44 55 66 77 88 99 00”	“Digit number incremented” to appear thirty times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
46	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“Hello. How are you? I'm great thanks! ##£$%&”	“Digit number incremented” to appear zero times	As expected

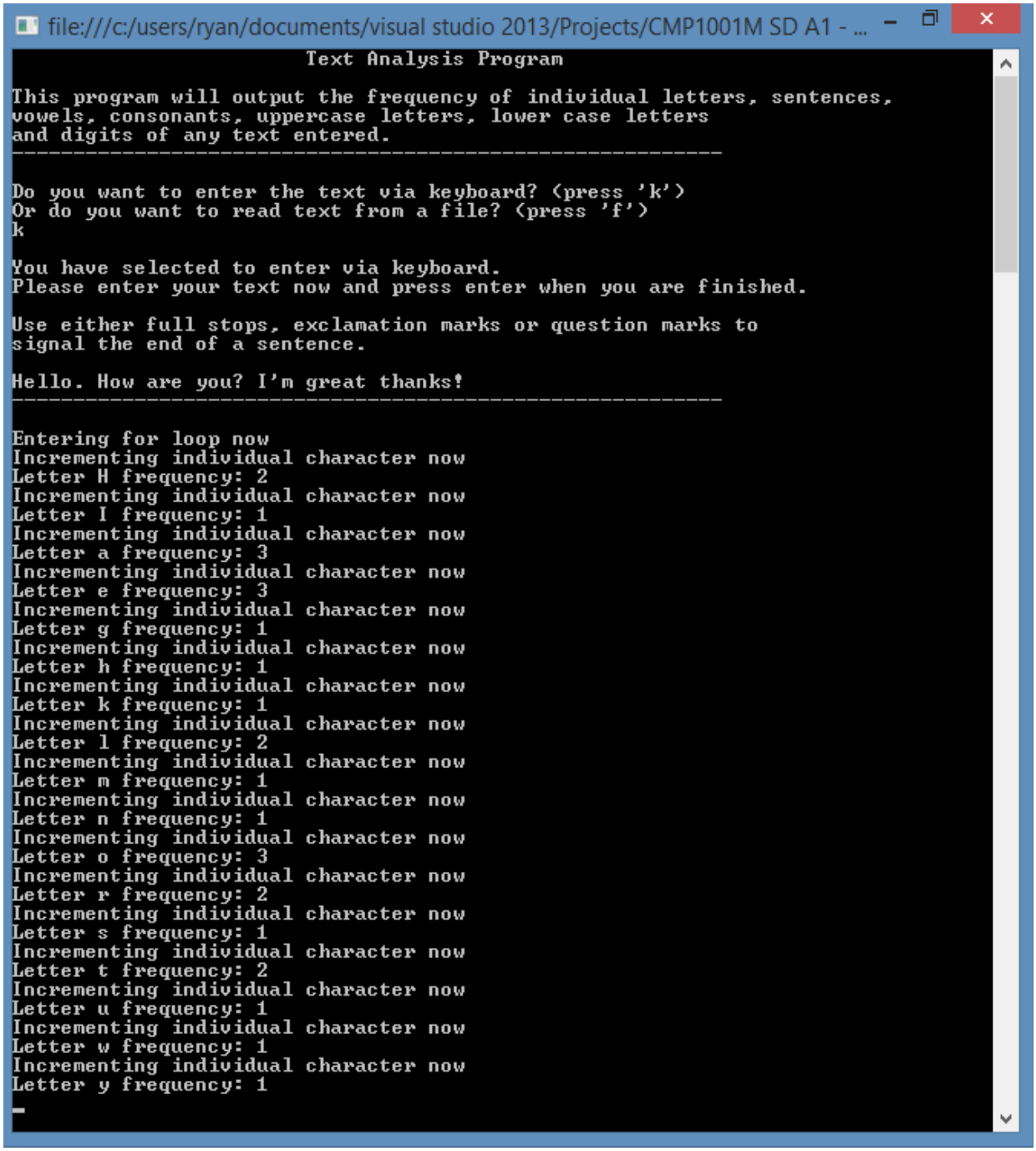


White Box Testing – LetterFrequency()

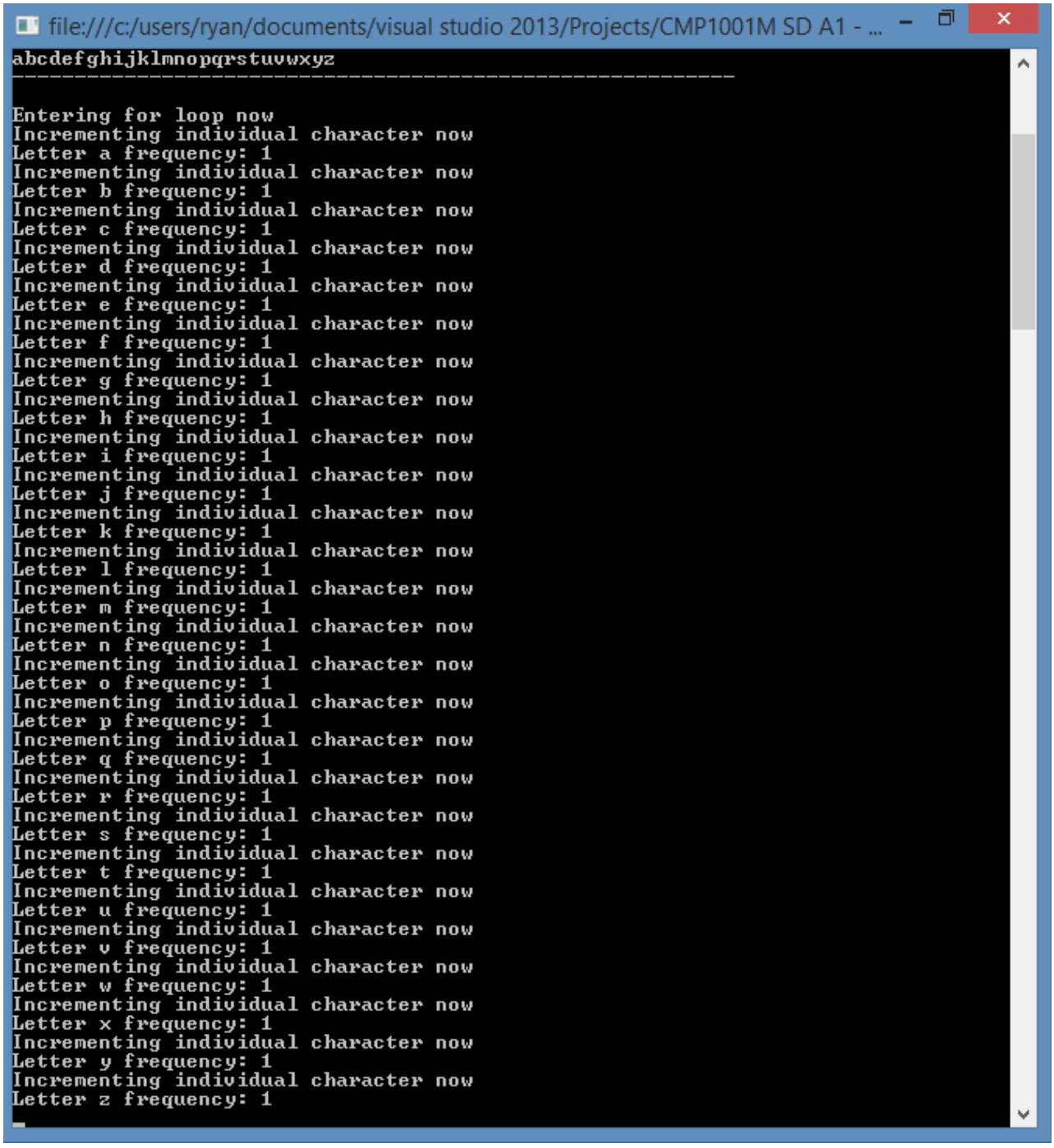
For this test, I will put a Console.WriteLine statement at the top of the for loop, then a Console.WriteLine every time an individual letter is added to the array.

```
1 reference
public static void LetterFrequency()
{
    {
        foreach (char t in Variables.text_entered)
        {
            Variables.frequency[(int)t]++;
        }
        Console.WriteLine("Entering for loop now");
        for (int i = 0; i < (int)char.MaxValue; i++)


            if (Variables.frequency[i] > 0 && char.IsLetter((char)i))

            {
                Console.WriteLine("Incrementing individual character now");
                Console.WriteLine("Letter {0} frequency: {1}", (char)i, Variables.frequency[i]);
            }
    }
}
```

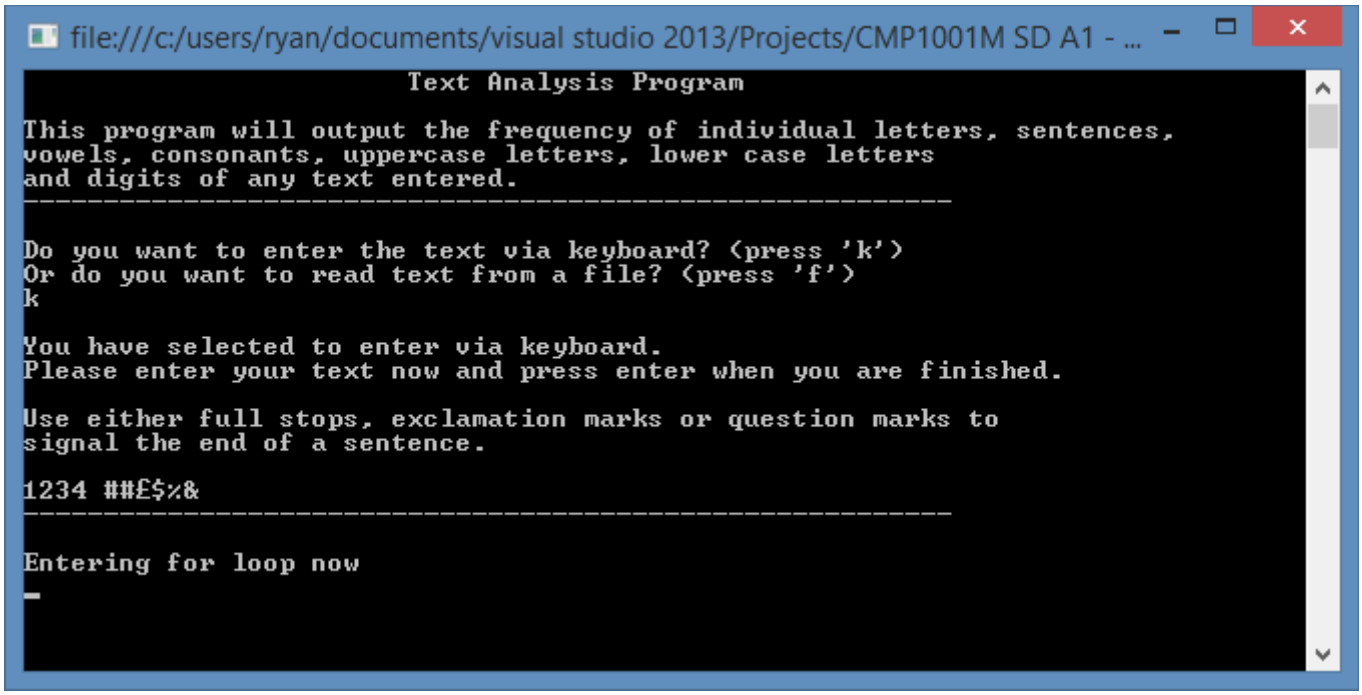
Test number	Purpose	Input	Expected Result	Actual Result
47	To verify that the Console.WriteLine statements are executed when valid data is entered	“Hello. How are you? I'm great thanks!”	“Entering for loop now” to appear once “Incrementing individual letter now” to appear seventeen times	As expected



Test number	Purpose	Input	Expected Result	Actual Result
48	To verify that the Console.WriteLine statements are executed when valid data is entered	“abcdefghijklmnopqrstuvqxyz”	“Entering for loop now” to appear once “Incrementing individual letter now” to appear twenty-six times	As expected



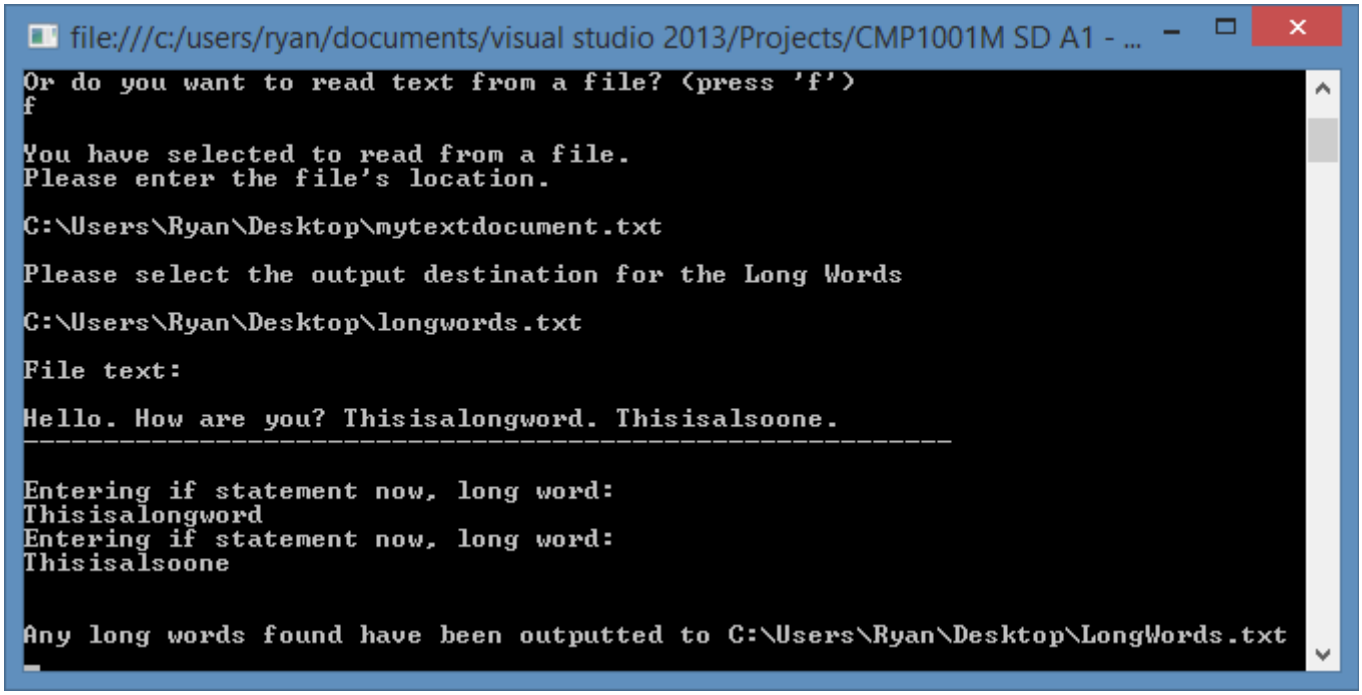
Test number	Purpose	Input	Expected Result	Actual Result
49	To verify that the Console.WriteLine statement isn't executed when the if statement is false	“1234 ##£$ %&”	“Entering for loop now” to appear once	As expected



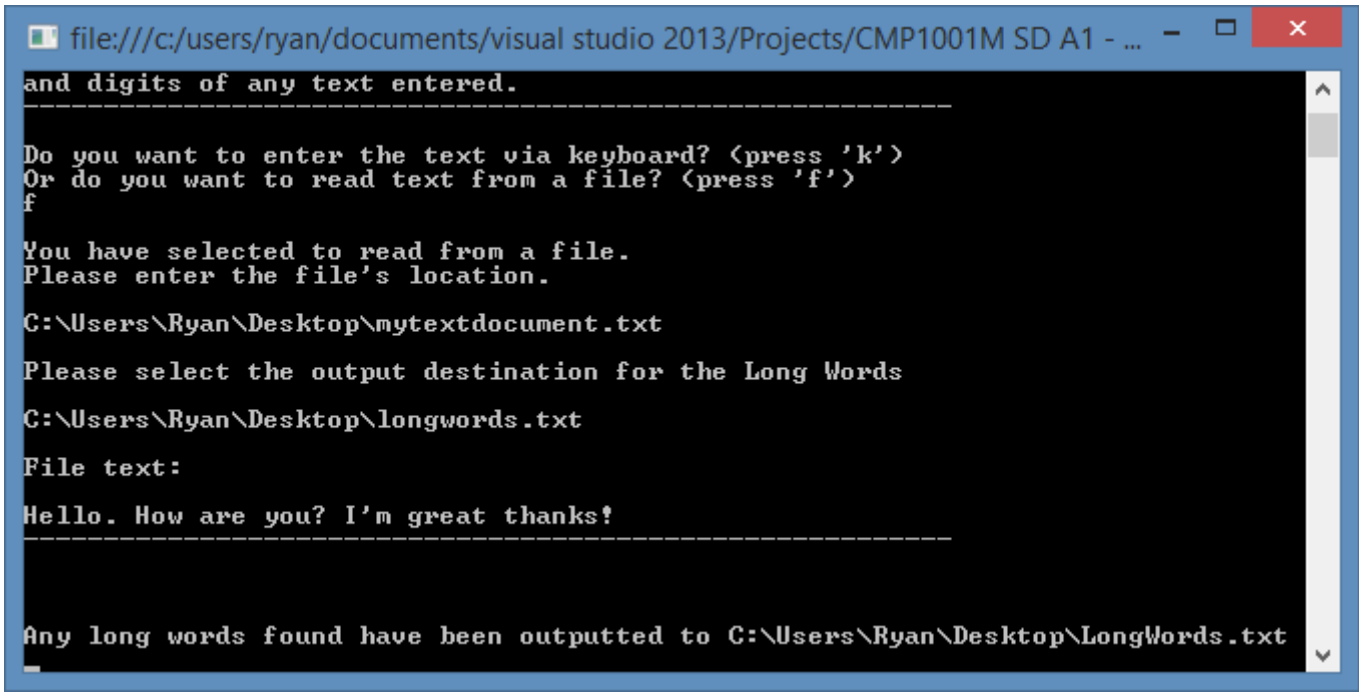
White Box Testing – LongWords()

For this test, I will enter a Console.WriteLine statement which will appear every time a long word is found.

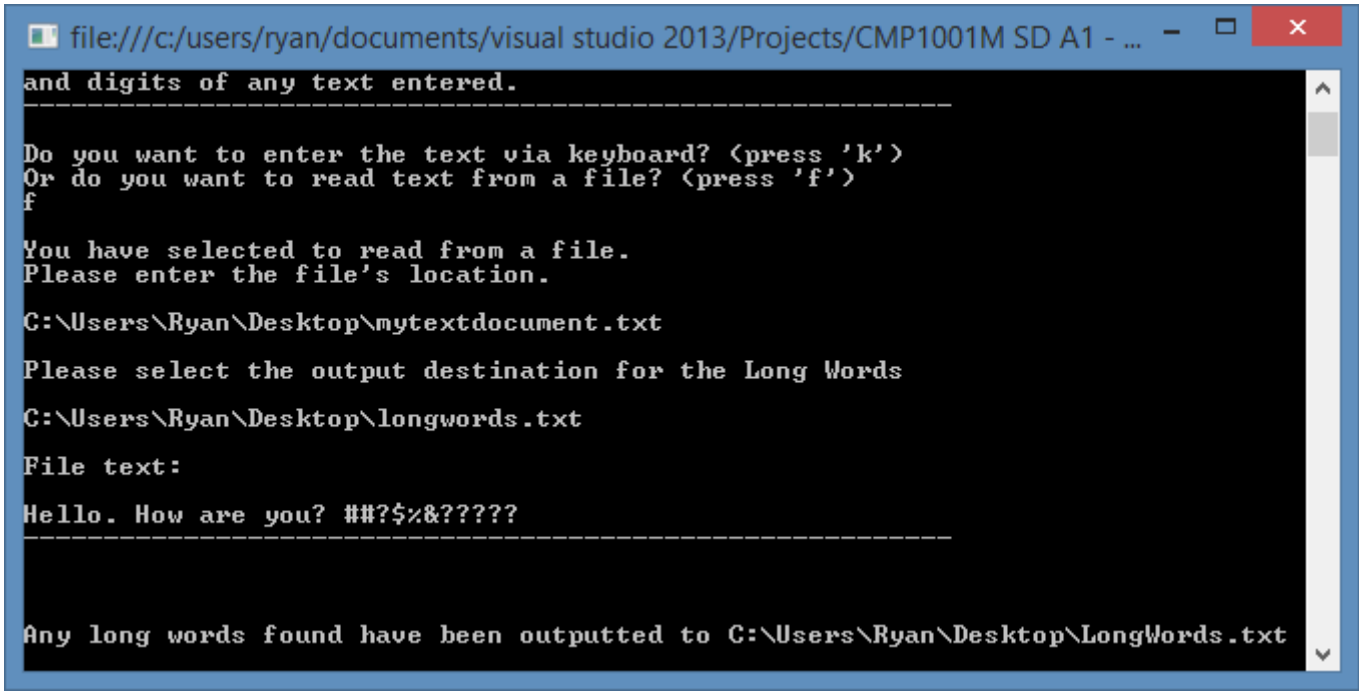
Test number	Purpose	Input	Expected Result	Actual Result
50	To verify the number of times the if statement has been executed	“Hello. How are you? Thisisalongword. Thisisalsoone”	“Entering if statement now, long word:” to appear twice	As expected



Test number	Purpose	Input	Expected Result	Actual Result
51	To verify that the Console.WriteLine statements are only executed when the if statement is true	"Hello. How are you? I'm great thanks!"	No Console.WriteLine statements	As expected



Test number	Purpose	Input	Expected Result	Actual Result
52	To verify that the Console.WriteLine statements are only executed when the if statement is true	“Hello. How are you? ##£$%&£££ ££”	No Console.WriteLine statements	As expected



Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Text.RegularExpressions;


namespace CMP1001M_SD_A1___Text_Anlysis_Program
{
    public static class Program
    {
        public static class MainClass
        {
            public static void Main()
            {
                //welcome text
                Console.WriteLine("\t\t\tText Analysis Program");
                Console.WriteLine("\nThis program will output the frequency of individual
letters, sentences,\nvowels, consonants, uppercase letters, lower case letters\nand digits of any
text entered.");
                Console.WriteLine("-------------------------------------------------------------");
                Console.WriteLine("\nDo you want to enter the text via keyboard? (press 'k') \nOr
do you want to read text from a file? (press 'f')");

                Variables.option_entered = Console.ReadLine();

                if (Variables.option_entered == "k" || Variables.option_entered == "K")
                {

                    Console.WriteLine("\nYou have selected to enter via keyboard. \nPlease enter
your text now and press enter when you are finished.");
                    Console.WriteLine("\nUse either full stops, exclamation marks or question
marks to \nsignal the end of a sentence.\n");
                    Variables.text_entered = Console.ReadLine();

Console.WriteLine("-------------------------------------------------------------\n");

                    //calls all of the methods used (except LongWords())
                    Calculations.LetterFrequency();
                    Calculations.NumOfSentences();
                    Calculations.NumOfConsonants();
                    Calculations.NumOfVowels();
                    Calculations.NumOfUpperCase();
                    Calculations.NumOfLowerCase();
                    Calculations.NumOfDigits();
                    Calculations.DisplayCalculations();
                    Console.ReadLine();

                }

                {
                    if (Variables.option_entered == "f" || Variables.option_entered == "F")
                    {
                        //exception handling
                        try
                        {
                            Console.WriteLine("\nYou have selected to read from a file. \nPlease
```

```
enter the file's location.\n");
                            Variables.text_entered = File.ReadAllText(Console.ReadLine());
                            Console.WriteLine("\nPlease select the output destination for the
Long Words\n");
                            Variables.long_words_output = Console.ReadLine();
                            Console.WriteLine("\nFile text: \n\n{0}", Variables.text_entered);

Console.WriteLine("------------------------------------------------------------\n");
                        }
                        //file not found is the only exception that can happen here
                        catch (Exception NoFile)
                        {
                            Console.WriteLine("\n" + NoFile.Message);
                            Console.WriteLine("Please try again\n");

Console.WriteLine("------------------------------------------------------------\n");
                            Main();
                        }
                        {
                            Calculations.LetterFrequency();
                            Calculations.NumOfSentences();
                            Calculations.NumOfConsonants();
                            Calculations.NumOfVowels();
                            Calculations.NumOfUpperCase();
                            Calculations.NumOfLowerCase();
                            Calculations.NumOfDigits();
                            Calculations.LongWords();
                            Calculations.DisplayCalculations();
                            Console.WriteLine("\n\nAny long words found have been outputted to "
+ @"C:\Users\Ryan\Desktop\LongWords.txt");
                            Console.ReadLine();
                        }
                    }

                    else Console.WriteLine("\nInvalid input, please try again\n");

Console.WriteLine("------------------------------------------------------------\n");
                    {
                        //using recursion for error handling - this will restart the program
                        Main();

                    }
                }
            }

            //I have created an individual class for all of the calculations that the program
will be carrying out
            public class Calculations
            {
                //method to calculate the number of sentences
                public static int NumOfSentences()
                {
                    foreach (char c in Variables.text_entered)
                    {
                        if (c == '.' || c == '?' || c == '!')
                        {

                            Variables.num_of_sentences++;
                        }
                    }
                    return Variables.num_of_sentences;
                }
```

```
        //method to count the number of vowels
        public static int NumOfVowels()
        {
            foreach (char v in Variables.text_entered)
            {


                if (v == 'a' || v == 'A' || v == 'e' || v == 'E' || v == 'i' || v == 'I'
                    || v == 'o' || v == 'O' || v == 'u' || v == 'U')
                {
                    Variables.num_of_vowels++;
                }
            }
            return Variables.num_of_vowels;
        }
        //method to count the number of consonants
        public static int NumOfConsonants()
        {
            foreach (char c in Variables.text_entered)
            {
                if (c == 'b' || c == 'B' || c == 'c' || c == 'C' || c == 'd' || c == 'D'
                    || c == 'f' || c == 'F' || c == 'g' || c == 'G' || c == 'h' || c ==
'H'
                    || c == 'j' || c == 'J' || c == 'k' || c == 'K' || c == 'l' || c ==
'L'
                    || c == 'm' || c == 'M' || c == 'n' || c == 'N' || c == 'p' || c ==
'P'
                    || c == 'q' || c == 'Q' || c == 'r' || c == 'R' || c == 's' || c ==
'S'
                    || c == 't' || c == 'T' || c == 'v' || c == 'V' || c == 'w' || c ==
'W'
                    || c == 'x' || c == 'X' || c == 'y' || c == 'Y' || c == 'z' || c ==
'Z')
                {

                    Variables.num_of_consonants++;
                }
            }
            return Variables.num_of_consonants;
        }

        //method for counting the number of uppercase letters
        public static int NumOfUpperCase()
        {
            for (int i = 0; i < Variables.text_entered.Length; i++)
            {
                //Counting the number of uppercase letters using the IsUpper function
                if (char.IsUpper(Variables.text_entered[i]))
                {
                    Variables.num_of_uppercase++;
                }
            }
            return Variables.num_of_uppercase;
        }

        //method for counting the number of lowercase letters
        public static int NumOfLowerCase()
        {
            for (int i = 0; i < Variables.text_entered.Length; i++)
            {
                //Counting the number of lowercase letters using the IsLower function
```

```
                    if (char.IsLower(Variables.text_entered[i]))
                    {
                        Variables.num_of_lowercase++;
                    }
                }
                return Variables.num_of_lowercase;
            }

            //method to cound the number of digits
            public static int NumOfDigits()
            {

                foreach (char d in Variables.text_entered)

                    if (char.IsDigit(d))
                    {
                        Variables.num_of_digits++;
                    }
                return Variables.num_of_digits;
            }

            //method for counting and outputting the frequency of each individual letter
            //technique inspired from http://www.dotnetperls.com/count-letter-frequencies


            public static void LetterFrequency()
            {
                {
                    foreach (char t in Variables.text_entered)
                    {
                        Variables.frequency[(int)t]++;
                    }
                    for (int i = 0; i < (int)char.MaxValue; i++)


                        if (Variables.frequency[i] > 0 && char.IsLetter((char)i))
                        {
                            Console.WriteLine("Letter {0} frequency: {1}", (char)i,
Variables.frequency[i]);
                        }
                }
            }

            //method to count the number of long words
            public static void LongWords()
            {
                //using Regular Expressions to strip the text entered of punctuation
                //[^\w\s] finds any character that is not a word or space character
                //I have then replaced these characters with "" (nothing)
                //technique inspired from http://stackoverflow.com/questions/5871793/remove-
punctuation-from-string-with-regex


                string trimmedWord = Regex.Replace(Variables.text_entered, @"[^\w\s]", "");

                //this splits the trimmed word up every time a space character is found and
then stores the substrings in an array

                string[] splitWord = trimmedWord.Split(' ');

                //checking the length of the invividual substrings
                foreach (string s in splitWord)
```

```
                    {
                        if (s.Length >= 7)
                        {
                            Console.WriteLine("Entering if statement now, long word:");
                            Console.WriteLine(s);
                            File.AppendAllText(Variables.long_words_output, s + " ");
                        }
                    }
                }


                //void method to display the results of the calculations
                public static void DisplayCalculations()
                {
                    Console.WriteLine("\nNumber of sentences = {0}", Variables.num_of_sentences);
                    Console.WriteLine("\nNumber of vowels = {0}", Variables.num_of_vowels);
                    Console.WriteLine("\nNumber of consonants = {0}",
Variables.num_of_consonants);
                    Console.WriteLine("\nNumber of upper case letters = {0}",
Variables.num_of_uppercase);
                    Console.WriteLine("\nNumber of lower case letters = {0}",
Variables.num_of_lowercase);
                    Console.WriteLine("\nNumber of digits = {0}", Variables.num_of_digits);
                }
            }

            //Declaring variables at in a public class so they can be easily used throughout the
program
            public static class Variables
            {
                //each individual variable is declared as public static

                public static int num_of_sentences = 0;
                public static int num_of_vowels = 0;
                public static int num_of_consonants = 0;
                public static int num_of_lowercase = 0;
                public static int num_of_uppercase = 0;
                public static int num_of_digits = 0;
                public static int[] frequency = new int[(int)char.MaxValue];
                public static string text_entered;
                public static string option_entered;
                public static string long_words_output;
            }
        }
    }
}
```

Bibliography

Stack Overflow. (2011) *Remove punctuation from sting with Regex.* [online] Available from: http://www.stackoverflow.com/questions/5871793/remove-punctuation-from-string-with-regex [Accessed 25 November 2013].

Dotnetpealrs. (Unknown year) *C# Count Letter Frequencies.* [online] Available from: http://www.dotnetperls.com/count-letter-frequencies [Accessed 2 December 2013].