This programming assignment tests ideas presented in sections 4, 5 and 6 of the course notes. You are advised to go over these sections carefully before attempting this assignment.

Program Specification:	**Word Count**

In the program you are expected to count the number of characters, words and lines in a given text input file. The character count encompasses all characters except the newline character.

For example, given the following file (where **…** represents a single space and ¿ represents a newline and Ξ represents end-of-file):

This … is … a ¿
… … ¿
¿
test X

Ignoring the newline characters, the character count is 15.

For the purposes of this program, words are defined as sequences of characters terminated by *gaps*. A *gap* is defined as one or more spaces or newlines. Any other character is considered a part of the word. Note that if the last word in a file is **not** terminated by a space or newline, it is not counted. In the above example, the last word "test" is terminated by the end-of-file marker, without any space or newline character after the word. Hence, the example has 3 words, rather than the intuitive answer of 4.

A line in the context of this assignment is *any* sequence of characters terminated by a newline. Sequences of newline characters count as different lines. In the above example, there are 3 lines. Note that if there were a newline after the last word "test", there would be 4 lines. The word "test" is not considered part of a line, but it is counted in the character count.

Note that the newline character in Windows based systems is represented by 2 characters '\r' and '\n'. Calling the get() function returns both characters separately, one at a time. Your program should **not** count '\r' and '\n' characters in the character count.

Such programs are used in the preliminary stage of a compiler called the parser. They are also used in word processors to give a word count summary of a text document, for instance the word count option under the Tools menu in Microsoft Word.

Input File Description

Your program must read its input from a file called **text.txt** – use of any other file name will result in a score of zero. The file contains ASCII text characters. You may assume that the data is logically correct, i.e. there are no binary characters in the data.

For instance, this is a sample input file

This … is … a ¿
… … ¿
¿
test X

You should not make any assumptions about the number of characters in the file. Your program should detect *end of input* and terminate correctly.

Program Description

Your program should read the input file and produce an output file, which contains the text of the input file with the following modifications:

1. All sequences (more than 1) of spaces in the input file are replaced by 1 space in the output file.
2. All sequences (more than 1) of newlines in the input file are replaced by 1 newline in the output file.
3. All other characters in the input file appear *as they are* in the output file.

In addition, based on the definitions above, your program should count the number of characters, words and lines in the program.

Output File Description

A sample output file produced from the sample input file above is shown below. The first line of your output should identify you by name, as shown. The second line should include the title "Word Count" only. The third line should be blank.

Your output file must be called **results.txt**. Use of any other file name will result in a score of zero.

The rest of the output file contains text from the input file after the appropriate modifications discussed in the preceding section. At the end of the output file put two newline characters and print out the number of characters, number of words and number of lines encountered in the input file. The numbers must be on separate lines.

Example output file: (for the above input)

Programmer: Srinidhi Varadarajan ¿
Word Count ¿
¿
This … is … a ¿
… ¿
test ¿
¿
Total characters: ¿
15 ¿
Total words: ¿
3 ¿
Total lines: ¿
3 ¿

Put two newline characters after the output text. Note that the input file (and hence the output file) does not have a newline character after the word "test. Hence, the first of the two newline characters appears after the word test.

Programming Standards:
You are expected to observe good programming/documentation standards. All the discussions in class about formatting, structure, and commenting your code should be followed. A copy of *Elements of Programming Style* is included with the course notes — if you don't have a copy we strongly suggest you read the on-line edition (available from the course web page). Some specifics:

- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, function, etc.
- Use named constants instead of variables where appropriate.
- Precede every major block of your code with a comment explaining its purpose.
- You must use indentation and blank lines to make control structures (like loops and if-else statements) more readable.

Hints:
This program is not as simple as it initially appears to be. Try to construct examples and figure out how your program should behave before you write any code. Pay special attention to the following instances:

- A word can be terminated by either a space or a newline. What happens to the word count when a word is terminated by 2 spaces and 2 newline characters? What happens to the line count?
    - Ans: Word count is incremented by 1. Line count is incremented by 2. You should confirm that your code achieves this result.
- In the preceding example, ensure that your program has the same effect on word count when a word is terminated by a newline.
- Sequences of space characters are replaced by a single space character. Sequences of newlines are replaced by a single newline. How should the sequence, “space newline space newline space newline” be replaced?
    - Ans: The sequence remains the same. Nothing is replaced. Remember a sequence is more than 1 character of the same type

Testing:
You should be certain that your program produces the output given above when you use the given input file. However, verifying that your program produces correct results on a single test case does not constitute a satisfactory testing regimen. You should test your program on all the posted input/output examples given along with this specification.