C# Assignment #1

You are given a list of international product codes that you need to sort. These product codes include a combination of numbers, punctuation, whitespace and other text. Each product code is on its own separate line. You are to produce a C# program that:

1. Takes two command line parameters. The first one is input file name, the second is the output file name.
2. The input file will be in UTF-8 encoding. The output file must also be in UTF-8 encoding.
3. Sort the product codes in the input file according the sorting rules listed below.
4. Place the sorted product codes into the output file, one per line, making no changes to the casing, spacing or punctuation of the product codes.
5. Use Linux style line endings in the output file (\n, not \r\n)
6. After the output file has been written, print to System.out the number of distinct product codes in the input file. “Distinct product codes” is defined below.

Product codes are:

* Product codes can be made up from a combination of digits, punctuation, whitespace, and non-digit characters.
* The digits are the normal Arabic numerals (0 thru 9)
* The non-digit characters are Latin, Cyrillic or Hiragana (Japanese).
* The punctuation characters are only “.”, “/”, and “-“.
* The whitespace characters are limited to “ “ (space).
* The punctuation and whitespace characters in a product code are considered “formatting” and are irrelevant to the value of the product code. (i.e. “A 1/C” is considered equivalent to “A1C” or “A/1-C” because the same patterns of digits and non-digit characters are present in all three.)
* Product codes are made up of alternating blocks of digits and non-digits. In the example “XYZ123ABC” there are 3 blocks: “XYZ”, “123” and “ABC”. The transition between blocks is marked by a change from digit to non-digit or vise versa.
* Product code blocks are never more than 8 digits or non-digits long.
* There are never more than 8 blocks in a product code.

Here are the sorting rules:

* Ignore all punctuation and whitespace when sorting. (i.e. “123-ABC-K” is considered equivalent to 123A B-CK.)
* Sort is done “block-wise”. When comparing two product codes, the first block of each is compared. Continue to compare blocks in order until you run out of blocks or find that two blocks aren’t considered equivalent. If prefix blocks all match, the shorter product code comes first in the sort. (i.e. “ABC123KE” comes after “ABC123”)
* Blocks that contain digits always sort before blocks that contain non-digits.
* Blocks that contain digits are compared using their integer value, lower values coming first. (i.e. the block “0019” comes after the block “7”.)
* Blocks that contain digits whose integer values are equivalent are considered equivalent (i.e. “0019” is equivalent to “19”.)
* Blocks of non-digits are compared as if they were upper case. (i.e. “A” and “a” are considered equivalent and “Z” comes after “a”.)
* Blocks of non-digits are compared character-wise until a non-matching character is found. That non-matching character determines which block sorts before the other. The results of Character.compare() determines the sort order of two non-matching characters.
* Shorter blocks with the same prefix sort before longer blocks. (i.e. “ABC” comes after “AB”)
* The sort should be “stable”. That is, equivalent product codes appear in output in the same order that they appeared in input.

Count of Distinct Product Codes:

* This is the number of groups of equivalent product codes in the file.
* As an example, a file that contained “AB/C1”, “A-b-c/1” and “aBc1” would have only 1 distinct product code because three product codes are considered equivalent.
* A second example: A file with “AB/C1”, “A-b-c/1”, “YYY”, and “yY/Y” would have only 2 distinct product codes because the first two product codes are considered equivalent to each other and the last two product codes are equivalent to each other.

Constraints:

* Use only the classes/libraries that come standard with Visual Studio

Example:

Input File:

ABC-123 Илиев

Abc123KE

00123

33

aB-C1.2.3/kE

abc-123/илиев

Output File:

33

00123

Abc123KE

aB-C1.2.3/kE

ABC-123 Илиев

abc-123/илиев

Prints:

“There are 4 distinct product codes in the file”