Assignment 01 LINQ

Note: Use ListGenerators.cs & Customers.xml

LINQ – Element Operators

1. Get first Product out of Stock

2. Return the first product whose Price > 1000, unless there is no match, in which case null is returned.

3. Retrieve the second number greater than 5

```
Int [] Arr = {5, 4, 1, 3, 9, 8, 6, 7, 2, 0};
```

LINQ – Aggregate Operators

1. Uses Count to get the number of odd numbers in the array

```
Int [] Arr = {5, 4, 1, 3, 9, 8, 6, 7, 2, 0};
```

2. Return a list of customers and how many orders each has.

3. Return a list of categories and how many products each has

4. Get the total of the numbers in an array.

```
Int [] Arr = {5, 4, 1, 3, 9, 8, 6, 7, 2, 0};
```

5. Get the total number of characters of all words in dictionary_english.txt (Read dictionary_english.txt into Array of String First).

6. Get the length of the shortest word in dictionary_english.txt (Read dictionary_english.txt into Array of String First).

7. Get the length of the longest word in dictionary_english.txt (Read dictionary_english.txt into Array of String First).

8. Get the average length of the words in dictionary_english.txt (Read dictionary_english.txt into Array of String First).

9. Get the total units in stock for each product category.

10. Get the cheapest price among each category's products

11. Get the products with the cheapest price in each category (Use **Let**)

12. Get the most expensive price among each category's products.

13. Get the products with the most expensive price in each category.

14. Get the average price of each category's products.

LINQ - Ordering Operators

1. Sort a list of products by name

2. Uses a custom comparer to do a case-insensitive sort of the words in an array.

```
String [] Arr = {"aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry"};
```

3. Sort a list of products by units in stock from highest to lowest.

4. Sort a list of digits, first by length of their name, and then alphabetically by the name itself.

```
string [] Arr = {"zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"};
```

5. Sort first by-word length and then by a case-insensitive sort of the words in an array.

```
String [] Arr = {"aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry"};
```

6. Sort a list of products, first by category, and then by unit price, from highest to lowest.

7. Sort first by-word length and then by a case-insensitive descending sort of the words in an array.

```
String [] Arr = {"aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry"};
```

8. Create a list of all digits in the array whose second letter is 'i' that is reversed from the order in the original array.

```
string [] Arr = {“zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine”};
```

LINQ – Transformation Operators

1. Return a sequence of just the names of a list of products.

2. Produce a sequence of the uppercase and lowercase versions of each word in the original array (Anonymous Types).

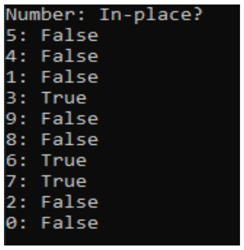
```
String [] words = {"aPPLE", "BlUeBeRrY", "cHeRry"};
```

3. Produce a sequence containing some properties of Products, including UnitPrice which is renamed to Price in the resulting type.

4. Determine if the value of int in an array matches their position in the array.

```
Int [] Arr = {5, 4, 1, 3, 9, 8, 6, 7, 2, 0};
```

Result

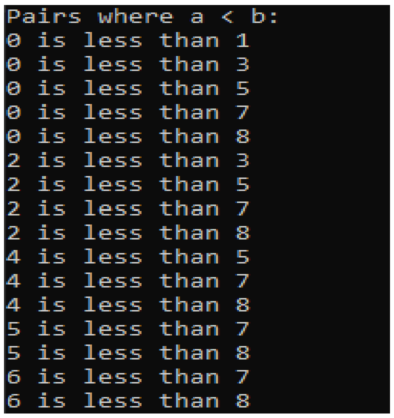


```
Number: In-place?
5: False
4: False
1: False
3: True
9: False
8: False
6: True
7: True
2: False
0: False
```

5. Returns all pairs of numbers from both arrays such that the number from numbersA is less than the number from numbersB.

int[] numbersA = { 0, 2, 4, 5, 6, 8, 9 };
int[] numbersB = { 1, 3, 5, 7, 8 };

Result



```
Pairs where a < b:
0 is less than 1
0 is less than 3
0 is less than 5
0 is less than 7
0 is less than 8
2 is less than 3
2 is less than 5
2 is less than 7
2 is less than 8
4 is less than 5
4 is less than 7
4 is less than 8
5 is less than 7
5 is less than 8
6 is less than 7
6 is less than 8
```

6. Select all orders where the order total is less than 500.00.

7. Select all orders where the order was made in 1998 or later.