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 – Set Operators

1. Find the unique Category names from Product List

2. Produce a Sequence containing the unique first letter from both product and customer names.

3. Create one sequence that contains the common first letter from both product and customer names.

4. Create one sequence that contains the first letters of product names that are not also first letters of customer names.

5. Create one sequence that contains the last Three Characters in each name of all customers and products, including any duplicates

LINQ – Quantifiers

1. Determine if any of the words in dictionary_english.txt (Read dictionary_english.txt into Array of String First) contain the substring 'ei'.

2. Return a grouped a list of products only for categories that have at least one product that is out of stock.

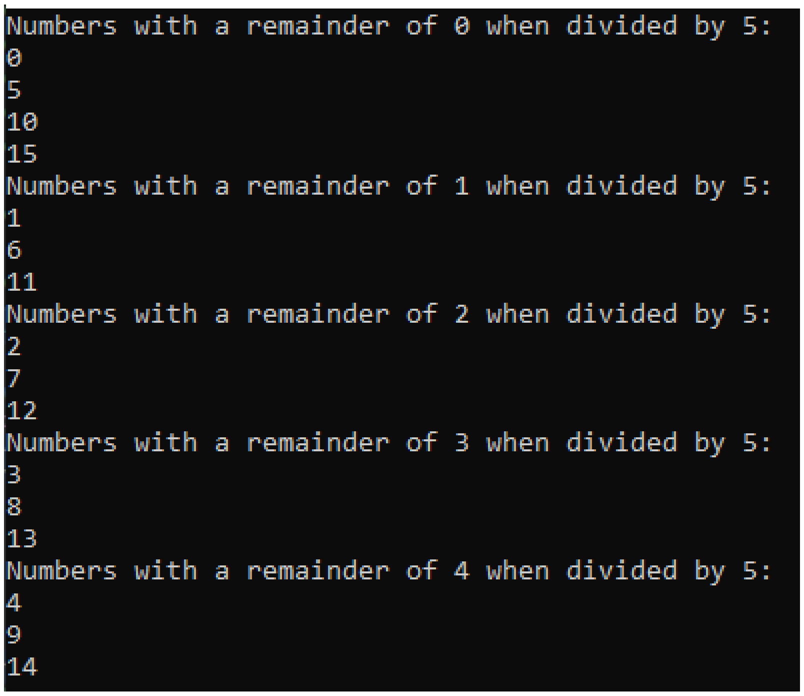
3. Return a grouped a list of products only for categories that have all of their products in stock.

LINQ – Grouping Operators

1. Use group by to partition a list of numbers by their remainder when divided by 5

```
List<int> numbers = new list<int> {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
```

Output:



```
Numbers with a remainder of 0 when divided by 5:
0
5
10
15
Numbers with a remainder of 1 when divided by 5:
1
6
11
Numbers with a remainder of 2 when divided by 5:
2
7
12
Numbers with a remainder of 3 when divided by 5:
3
8
13
Numbers with a remainder of 4 when divided by 5:
4
9
14
```

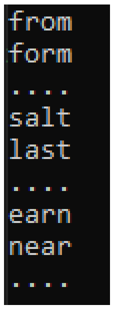
2. Uses group by to partition a list of words by their first letter.
   Use dictionary_english.txt for Input

3. Consider this Array as an Input

```
String [] Arr = {"from", "salt", "earn", " last", "near", "form"};
```

Use Group By with a custom comparer that matches words that are consists of the same Characters Together

Output:



```
from
form
....
salt
last
....
earn
near
....
```