Dynamic Linq

This assignment will give you practical experience with **Linq queries in dynamic environments** (e.g. when processing web data).

For a general view on **Dynamic Linq**, see this doc, esp. the introduction and the section "**IQueryable Extension Methods**":

https://github.com/kahanu/System.Linq.Dynamic/wiki/Dynamic-Expressions

> Database applications frequently rely on "Dynamic SQL" — queries that are constructed at run-time through program logic.

> The LINQ infrastructure supports similar capabilities through dynamic construction of expression trees using the classes in the System.Linq.Expressions namespace.

> Expression trees are an appropriate abstraction for a variety of scenarios, but for others a **string**-based representation may be more convenient.

Briefly, here we want to run a simple **subset** of Linq queries expressed as **strings**, not as statically complied lambdas, nor even expression trees.

From all the methods supported by Dynamic Linq, we will further restrict ourselves to only the following sublist: **Select**, **Where**, **OrderBy**, **Take**, and **Skip** (in their simplest form, without the possibile extra parameters).

Our **starting sequence** will always be a **string sequence**.

The input is read from **stdin** (as with the previous two assignments) and has the following structure:

- **First line** gives the **starting sequence**: **printable words** in free format, separated by **one or more spaces**.
- **Each subsequent line** indicates a required **transformation**, using one the mentioned **Dynamic Linq** methods.

  It is a sequence of **ASCII words**, separated by one or more spaces:
  - The **first** word is the **name** of the **Dynamic Linq method**
  - The **rest** of the line is a string representation of the **argument** for the **method** (recall that we do not use additional parameters).

The output to **stdout** must contain the final **resulting sequence**, after applying, **in the given order**, **all** the mentioned transformations, as **printable words**, separated by **single spaces**.

All sequences, including the final one, will consist of only **simple** items: **strings**, **numbers**, **booleans** (no nested subsequences or complex objects). Strings are printed without surrounding quotes.

We will use the compiled **DynamicLinqUoA.dll** given in the **A7 samples folder**. Note that these methods apply only to sequences that are type-casted as **IQueryable** (b/c internally they build … expression trees).

The rest of this handout presents and briefly explains the given **samples**. You have to transform the starting sequences using **Dynamic Linq**.

You need **not** bother with the **explanation** how these dynamic queries are further transformed to statically compiled Linq (this is the library's job, not ours).

Note that:

- o **Select**, **Where**, and **OrderBy** take a **string** parameter (unlike static Linq, where they take a Func or Expression parameter, often a lambda)
- o **Take** and **Skip** take an **int** parameter (as in static Linq)
- o See the **Appendix** for their DynamicLinq signature
- o **it** is the **DynamicLinq** name of the **current** lambda parameter (see following examples).

Test case #1

Input – stdin, redirected from input1.txt:

```
  the quick brown fox   jumps over the lazy dog
Where Length < 4
```

DynamicLinq pipeline:

```
.Where ("Length < 4")
```

Explainer - equivalent static Linq pipeline:

```
.Where (it => it.Length < 4)
```

Final result – output1.txt, redirected from stdout:

```
the fox the dog
```

Test case #2

Input – stdin, redirected from input2.txt:

```
  the quick brown fox   jumps over the lazy dog
Where Length < 4
Select it
```

DynamicLinq pipeline:

```
.Where ("Length < 4") .Select ("it")
```

Explainer - equivalent static Linq pipeline:

```
.Where (it => it.Length < 4) .Select (it => it)
```

Final result – output2.txt, redirected from stdout:

```
the fox the dog
```

Test case #3

Input – stdin, redirected from input3.txt:

```
  the quick brown fox   jumps over the lazy dog
Where Length < 4
Select Length
```

DynamicLinq pipeline:

```
.Where ("Length < 4") .Select ("Length")
```

Explainer - equivalent static Linq pipeline:

```
.Where (it => it.Length < 4) .Select (it => it.Length)
```

Final result – output3.txt, redirected from stdout:

```
3 3 3 3
```

Test case #4

Input – stdin, redirected from input4.txt:

```
  the quick brown fox   jumps over the lazy dog
OrderBy Length DESC, it ASC
```

DynamicLinq pipeline:

```
.OrderBy ("Length DESC, it ASC")
```

Explainer - equivalent static Linq pipeline:

```
.OrderByDescending (it => it.Length) .ThenBy (it => it)
```

Final result – output4.txt, redirected from stdout:

```
brown jumps quick lazy over dog fox the the
```

Test case #5

Input – stdin, redirected from input5.txt:

```
  the quick brown fox   jumps over the lazy dog

OrderBy it, Length
```

DynamicLinq pipeline:

```
.OrderBy ("it, Length")
```

Explainer - equivalent static Linq pipeline:

```
.OrderBy (it => it) .ThenBy (it => it.Length)
```

Final result – output5.txt, redirected from stdout:

```
brown dog fox jumps lazy over quick the the
```

Test case #6

Input – stdin, redirected from input6.txt:

```
  the quick brown fox   jumps over the lazy dog

Skip 4

Take 2
```

DynamicLinq pipeline:

```
.Skip (4) .Take (2)
```

Explainer - equivalent static Linq pipeline:

```
.Skip (4) .Take (2)
```

Final result – output6.txt, redirected from stdout:

```
jumps over
```

Programs:

(A#7) a C# solution (required).

Essentially, each program must be totally contained in **one single file** and use only standard libraries extant in the labs, plus **DynamicLinqUoA.dll** (that will also be available on **automarker**).

- o The input must be read from **stdin**, i.e. Console.In in C#.
- o The output must be written to **stdout**, i.e. Console[.Out] in C#.

Submission

Please submit to the **automarker**:

https://www.automarker.cs.auckland.ac.nz/student.php

APPENDIX – DynamicLinq signatures (w/o optional extra parameters)

```
IQueryable Where (this IQueryable source, string predicate)

IQueryable Select (this IQueryable source, string selector)

IQueryable OrderBy (this IQueryable source, string ordering)

IQueryable Take (this IQueryable source, int count)

IQueryable Skip (this IQueryable source, int count);
```

Please do not forget:

Add the **Dynamic Linq namespace**:

```
using System.Linq.Dynamic;
```

Include library **DynamicLinqUoA.dll** in the compilation

```
csc  -r:DynamicLinqUoA.dll  a7.cs
```

This library should reside in the **same folder** as your C# source and the compiled executable.