Warmup

This is the first assignment for Part II, which will familiarise you with the automarker and with our programming languages, style, and tools.

The assignment covers 4 course marks and additionally offers 3 possible bonus marks:

- **Base part, C#: 4 marks**
- Bonus part, F#: 1 mark
- Bonus part, Node.js: 1 mark
- Bonus part, Python 3: 1 mark

The C# part allocates 2 marks for completing all automarker tests and 2 marks for using the indicated functional style (partial marks possible for programs that do not meet the functional criteria).

All bonus marks require both the completion of all automarker tests and using the required functional style (no marks will be awarded for partial fulfilment of these tasks).

The functional style required here mandates the use of **higher-order functions**, while avoiding any explicit classical control constructs from the **for**/**while** family. Note that comprehensions, although also good style, are also prohibited here, to ensure that we focus on higher-order functions.

Specific requirements:

- C#: Use a .NET Core 3.1 .csproj project
  - System.IO methods;
  - System.Linq methods, such as Select().
- F#: Use a .NET Core 3.1 .fsproj project
  - System.IO methods (as in C#);
  - Functions of the standard core modules, such as Seq.map, Array.map, List.map (i.e. no LINQ for F#).
- Node.js: Use Node.js 12.X
  - IO functions from module fs;
  - Built-in array functions, such as map;
  - *optionally*, functions from package ramda.
- Python 3: Use Python 3.X
  - IO functions from module sys;
  - Built-in array functions, such as map;
  - *optionally*, functions from modules functools, and operator.

You can start using the provided skeletal frames. Please ask us if you want to use something that is not listed here, as it may not be available on automarker.

Problem: Start with a text supposed to contain int32 numbers in free format, i.e. possibly signed integer numbers in the standard 32 bits range.

If the input contains anything else, then print an error message and stop - more details later.

If there is no error, then proceed. Divide by two the even numbers. Remove duplicates. Print the sum of remaining items and stop.

This assignment uses the **standard input/output** conventions for the **automarker**.

Input: The input text file is given on **stdin** - Console.In in C# - by command-line redirection (**<**).

Output: Write on **stdout**- - Console.In in C#. In our case, write one single line, containing just the required number (no spaces).

Exceptions should be caught and their message should be printed on **stdout** (in lieu of the sum), prefixed by three stars and one space (***)

A few examples will help clarify the specs.

Example 1:

Test input:

```
1 2 3 4 5 6 7 8 9

  1   2   3   4   5   6   7   8   9
```

Input numbers:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Even numbers halved:

```
1, 1, 3, 2, 5, 3, 7, 4, 9, 1, 1, 3, 2, 5, 3, 7, 4, 9,
```

Duplicates removed:

```
1, 3, 2, 5, 7, 4, 9,
```

Expected output:

```
31
```

Example 2:

Test input:

```
1 2 3 4 5 6 7 8 9

  10   11   12



10 20

-100 +100
```

Expected output:

```
58
```

Example 3:

Test input:

```
1 2 3 4 5 6 7 8 9

  10  11  12  13  14  15  16  17  18  19

1 2 3 4 5 6 7 8 9

  10  11  12  13  14  15  16  17  18  19
```

Expected output:

```
120
```

Example 4:

Test input:

```
11  xyz  13
```

Expected output (C#):

```
*** Input string was not in a correct format.
```

In fact, any string that does not represent a 32 bits integer shall raise errors:

```
xyz  12xyz  3.14  9999999999999999999999999999
```

The texts of the **error messages may be specific** for each platform (C#, F#, Node.js, Python 3).

Submission:

One single source file, to the **automarker** (https://www.automarker.cs.auckland.ac.nz/student.php):

- **C# base part (required)**: Your .cs source file.
- F# bonus: Your .fs source file.
- Node.js bonus: Your .js source files.
- Python 3 bonus: Your .py source file.

Notes for .NET Core (C# and F#):

- The required project files will be automatically provided by automarker, so don't worry about these.

Expected error messages

Some error messages are standard (**S**), i.e. automatically generated by the parsing API; others need to be manually (**M**) crafted. The three stars plus space prefix is added by you.

C#, F#: all error messages are standard

```
xyz  12xyz  3.14

      *** Input string was not in a correct format.             (S)

9999999999999999999999999999

      *** Value was either too large or too small for an Int32.  (S)
```

Node.js: all error messages are custom (manual)

```
xyz  12xyz  3.14

      *** NaN                                                    (M)

9999999999999999999999999999

      *** NanInt32                                               (M)
```

Python 3: all error messages are standard, except one

```
xyz

      *** invalid literal for int() with base 10: 'xyz'          (S)

12xyz

      *** invalid literal for int() with base 10: '12xyz'        (S)

3.14

      *** invalid literal for int() with base 10: '3.14'         (S)

9999999999999999999999999999

      *** NanInt32                                               (M)
```

Thus, **attention** for the manually crafted messages (M), i.e. all Node.js and one case for Python 3.

Suggested readings (not all necessary!!)

C#:

Namespaces System; System.Linq (Select, …)

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

https://docs.microsoft.com/en-us/dotnet/csharp/linq/linq-in-csharp

https://www.dotnetperls.com/linq

F#:

Namespace System; Module Seq (map, …); *and/or* Array, List, …

https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/sequences

https://fsharp.github.io/fsharp-core-docs/reference/fsharp-collections-seqmodule.html

https://fsharpforfunandprofit.com/

https://www.tutorialspoint.com/fsharp/fsharp_sequences.htm

Node.js:

Module fs (File System)

Lambdas (aka arrow functions), Array methods (map, …); *optionally* ramda

https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d

https://fr.umio.us/why-ramda/

https://ramdajs.com/docs/

Python 3:

Module sys (System)

Lambdas, built-in functions (map, …); *optionally* itertools, functools, operator

https://www.geeksforgeeks.org/functional-programming-in-python/

https://www.w3schools.com/python/ref_func_map.asp

https://docs.python.org/3/tutorial/controlflow.html#lambda-expressions

https://docs.python.org/3/library/functional.html