David Šiška School of Mathematics, University of Edinburgh

2018/19 Semester 1

Object Oriented Programming with Applications

Problem Sheet 1 - Wednesday 26th September 2018[1]

Hints and common problems:

- When starting a new project (`File -> New -> Project...`) make sure you have "Visual C#" selected in the left panel of the window.
- There's no need to create a new solution for each exercise, best practice would be to work on each of them in a separate method and call these as needed from `Main` (comments can be used to temporarily "disable" some part of the code).
- You can't have methods defined within methods: methods live inside classes, so they must be contained directly between the two { } defining the scope of the class you're working with
- String formatting: the syntax for basic string formatting is
  `string myString = string.Format("This {0} how you {1} a string. It {0} easy as {2}-{3}-{4}!", "is", "format", 1, 2, 3);`
  where the enumeration in { } corresponds to the order in which the subsequent arguments are listed. Calling `Console.WriteLine(myString);` next will print: "This is how you format a string. It is easy as 1-2-3!" to the console.

Exercise 1.1. Let $x = 10864$, $y = 18817$. Write C# code to calculate:

$$w_1 = 9x^4 - y^4 + 2y^2,$$
$$w_2 = (3x^2 - y^2)(3x^2 + y^2) + 2y^2,$$
$$w_3 = (9x^4 + 2y^2) - y^4$$

first using the type `double` and display the output.

Exercise 1.2. Write C# code that will print "C# is easy." 100 time to the Console.

Exercise 1.3. Evaluate (using pen and paper, not computer) the sum:

$$S = 10^8 + \sum_{i=1}^{10^7} 10^{-10}.$$

Write C# code to evaluate the above expression using a `for` loop when evaluating the sum. Is your hand calculated (correct) answer close to the C# answer?

Exercise 1.4. The Fibonacci sequence is defined by the recurrence relation

$$F(n) = F(n-1) + F(n-2), \;\; n \geq 2, \;\; F(0) = 0, \, F(1) = 1;$$

Write C# code method (function) that will calculate the nth term.

[1] Last updated 19th September 2018

Exercise 1.5. Modify the sorting method from the lecture to sort the numbers from the largest to the smallest.
What is the number of comparisons the algorithm will perform in the *worst case* for an array with n elements?