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]

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.

Solution:

```
double x = 10864; double y = 18817;
double w1 = 9 * Math.Pow(x, 4) - Math.Pow(y, 4) + 2 * Math.Pow(y, 2);
double w2 = (3 * Math.Pow(x, 2) - Math.Pow(y, 2)) *
    (3 * Math.Pow(x, 2) + Math.Pow(y, 2)) + 2 * Math.Pow(y, 2);
double w3 = (9 * Math.Pow(x, 4) + 2 * Math.Pow(y, 2)) - Math.Pow(y, 4);
Console.WriteLine("w1 = {0}, w2 = {1}, w3 = {2}", w1, w2, w3);
```

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

```
for (int i = 0; i < 100; i++)
{
    Console.WriteLine("C# is easy!");
}
```

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?
Solution:

```
double s = 1e8;
for (int i = 0; i < 1e7; i++)
{
    s += 1e-10;
}
Console.WriteLine(s);
```

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.
Solution:

[1] Last updated 9th October 2018

```
static int Fibonacci(int n)
{
    if (n == 0 || n == 1)
    {
        return n;
    }
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

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?
Solution:

```
static void MySort(double[] numbers)
{
    bool swapped;
    do
    {
        swapped = false;
        for (int i = 0; i < numbers.Length - 1; i++)
        {
            if (numbers[i] < numbers[i + 1])
            {
                double tmp = numbers[i];
                numbers[i] = numbers[i + 1];
                numbers[i + 1] = tmp;
                swapped = true;
            }
        }
    }
    while (swapped);
}
```

The worst-case complexity is $\mathcal{O}(n^2)$