Lab 1.2 - .NET command line compiler tools and IL exercise

1. Open notepad and enter this code:

```
class Calc
{
    static int Add(int a, int b)
    {
        return a + b;
    }
}
```

2. Save the file as `add.cs`
    a. Make sure that notepad does not add the txt extension to the file
3. Open Visual Studio command window
    a. This provides a development environment where all the .NET SDK tools are under the search path
    b. Compile the code to a module using the C# compiler:
    `csc /target:module add.cs`
    c. Make sure you've got a file named: `add.netmodule`
    d. Open the `add.netmodule` using the `ILDASM` tool:
        i. `ILDASM add.netmodule`
            1. Look at the MANIFEST
            2. Look at the `Calc` class and the Add method
                a. Explain what each of IL line of code does
    e. Now, dump the IL code to a file using the `ILDASM` tool:
        i. `ILDASM add.netmodule /out=calc.il`
        ii. You have got two files; can you explain what do they contain?
        iii. Open the result file using notepad:
            1. `notepad calc.il`
        iv. Change the name of the module to be `calc.netmodule`:
            1. `.module add.netmodule` ➔ `.module calc.netmodule`
        v. Copy the Add method, and create a new function named `Subtract`
            1. Change the function name to be `Subtract`
            2. Change the add IL opcode to a sub IL opcode
            3. Change the comment: // end of method `Calc::Subtract`
        vi. Save the `calc.il` file
    f. Compile the IL file using:
        i. `ilasm /DLL calc.il /output:calc.netmodule`
        ii. The result should be a `calc.netmodule`
            1. Open the new file using `ILDASM`
            2. You should see the new `Subtract` function
    g. Open a new text file in notepad:
        i. `notepad program.cs`

ii. Enter the following code:

```
using System;

class Program
{
	static void Main(string [] argv)
	{
		if (argv.Length != 2)
		{
			Console.WriteLine("Use: Calc [number]
[number], for example: Calc 10 6");
			return;
		}
		try
		{
			int x = int.Parse(argv[0]);
			int y = int.Parse(argv[1]);

			Console.WriteLine($"{x} + {y} =
{Calc.Add(x,y)}");
			Console.WriteLine($"{x} - {y} =
{Calc.Substract(x,y)}");
		}
		catch(Exception ex)
		{
			Console.WriteLine(ex.Message);
		}
	}
}
```

iii. Compile the code using `csc program.cs /out:calc.exe`
   1. Can you explain the error?

iv. Try to compile the code again using:

```
csc program.cs /addmodule:calc.netmodule /out:calc.exe
```

   1. Can you explain the errors?

v. We forgot to make the methods Add and `Subtract` public methods.

vi. Instead of going back to the original C# file, we can edit the `calc.il` file.
   1. Open `calc.il` in notepad and make the Add and `Subtract` method public by replacing the private attribute to a public:

```
.method private hidebysig static int32
          Add(int32 a,
              int32 b) cil managed
```

è

```
.method public hidebysig static int32
Add(int32 a,
int32 b) cil managed
```

vii. Recompile the `calc.il` file:
   1. `ilasm /DLL calc.il /output:calc.netmodule`

viii. Recompile the `program.cs` file:
   1. `csc program.cs /addmodule:calc.netmodule /out:calc.exe`

ix. Try the program:
   1. `Calc 4 2`
   2. `Calc`
   3. `Calc 5`
   4. `Calc a b`

x. Open `Calc.exe` using `ildasm`:
   1. `Ildasm calc.exe`
   2. Press `Ctrl-M`
   3. Can you explain what you see?

4. Debugging IL code:
   a. Dump `calc.exe` IL using:
      i. `Ildasm calc.exe /out:program.il`
   b. Compile the il code:
      i. `ilasm /debug /exe /resource=program.res program.il /output=calc.exe`
   c. start Visual Studio:
      i. `devenv calc.exe`
   d. Right-Click the calc project under the solution and choose properties
      i. Add 4 2 as the arguments

      ii. Save the change (`Ctrl-s`)
      iii. Right click once again on the `calc.exe` and choose Debug:

      iv. Use F10 to single step over the IL code
5. Read this for deeper understanding:
   a. https://blogs.msdn.microsoft.com/junfeng/2005/02/12/netmodule-vs-assembly/