CSCM985, Lab Dafny-3

To be ticked off:

- on **20.11.2025** or
- a week later, on **27.11.2025** at the beginning of the lab.

Please work in teams of two. There are Computer Instructions at the begin of this Lab Sheet.

- There are is on task with three sub-items.
- There is one challange task.
- All group participants need to be present to be ticked off.
- Check your marks on Lab-Tracker after having been ticked off, marks cannot be changed after.

Learning Outcomes

- ◎ Modeling in Dafny
- ◎ Development of proven to be correct programs in Dafny

Where to Look Up Things

- 👁 Dafny Cheat Sheet: `https://dafny.org/latest/DafnyCheatsheet.pdf`

Computer Instructions

→ sign in to OneDrive in order to have your files synched to the cloud

→ Under Teaching Software
Faculty
choose VScode [Do not choose Visual Studio!!!]
start Visual Studio Code (it might be the case that it still needs to be installed)
note: the window is often hidden

→ In VScode choose extensions (left sidebar)
enter Dafny
select Dafny
click install
create a new file with extension .dfy, say hugo.dfy
click into this file
if there are migration instruction [likely], accept the update to version 4.6.0.0, and you are ready to go

→ In case there is no migration pop-up
click on Dafny in the extensions field
click uninstall
click install
(possibly repeat this cycle)
until you are offered restart extension
click restart extension
a pop-up upgrade version appears accept the upgrade to 4.6.0

→ click on hugo.dfy
check for Dafny 4.6.0.0 in the bottom right corner - then things are correctly installed

→ Seeing Dafy in Action:

Download the file hello.dfy
Open it in VScode
Press F5: this executes hello.dfy
– you should see "hello" and the number "34" on the screen
Change the assertion "assert 42 > 0;" to "assert 42 < 0;"
– you see a right marker on the left of this line

Task 1 - First Steps on the Way of Becoming a “Dafny Guru”

Introduction to the Task:

After studying many pre-compiled examples, it is time for you to work independently with Dafny. This means carrying out the following three steps:

- Modelling: representing the natural language description as a method declaration and a contract in Dafny.
- Coding: writing a suitable method body in Dafny that computes the required return values.
- Annotating: (if need be) writing assert statements in order to understand what holds at different points in your program and/or writing invariant statements that support Dafny in proving your code to be correct.

➜ Write a Dafny method that takes two integers as parameters and then returns the distance (absolute value of the difference). Your method needs to have a contract, which Dafny proves to be valid.

➜ Write a Dafny method that takes three integers as parameters and returns 1 if they are in increasing order, -1 if they are in decreasing order, and 0 otherwise. Here, “increasing” means strictly increasing, with each value larger than its predecessor.

Note: When developing this code it will be useful to include assert statement concerning what knowledge has been established by the different tests, for instance `assert` $x \geq y$.

➜ Write Da Dafny method that takes an array of integer inputs as its parameter and returns the sequence of all adjacent duplicates. For example, if the input is 1 3 3 4 5 5 6 6 6 2, the program should print 3 5 6. Your method needs to have a contract, which Dafny proves to be valid.

Note: Dafny has a datatype `seq` which ought to be the return type of your method. You might need to make some experiments with this type: how to use it in formulae so that you can express your contract, e.g., how to uses the test `in` with a sequence, and also how compose sequences with the concatenation operator `+`.

Assessment Check List

✔ Three items to be ticked off.

Challeng Task – An Actual Challange

Introduction to the Task: Often, it is not clear of how to write a contract in Dafny: it is hard to find a “closed” mathemetical formula such as

```
 exists i: int :: (0 <= i < a.Length && a[i] == k)  ==> index == i
```

in order to express an expectation or a guarantee. One way out of this is to write a functions is Dafny such as

```
function factorial(n:nat):nat{
  if n <= 1 then 1
  else factorial (n-1)*n
}
```

Such functions extend the Dafny vocabulary and can be used in contracts

```
method it_factorial(n:nat) returns (r:nat)
  ensures r == factorial(n)
{ ...}
```

Such methodology bears two questions:

1. Can we have trust that the function has been correctly written down, i.e., is `factorial` correct?
2. If we have a function doing the job, why would we still need a method?

The answer to the first question is: we can trust the function definition to be correct if it is “easy” – i.e., one could spot a mistake immediately. The answer to the second question is twofold: (1) the iterative computation is often more efficients. (2) Usually, the contract for a realistic method will involve several newly defined functions, i.e., the method is not identical with just one function.

Overall, this discussion hints at an important topic: while Dafny

- provides a Turing complete programming language (one can’t ask for more) which is somehow ‘nice’ for programming and
- is ‘good’ at program verification,

one can have doubts on the question how good Dafny is a specification language.

Explanation - Roman Number System

The Roman number system has digits

I	1
V	5
X	10
L	50
C	100
D	500
M	1,000

Numbers are formed according to the following rules:

- Only numbers up to 3,999 are represented.
- As in the decimal system, the thousands, hundreds, tens, and ones are expressed separately.
- The numbers 1 to 9 are expressed as

I	1
II	2
III	3
IV	4
V	5
VI	6
VII	7
VIII	8
IX	9

  As you can see, an I preceding a V or X is subtracted from the value, and you can never have more than three I's in a row.
- Tens and hundreds are done the same way, except that the letters X, L, C and C, D, M are used instead of I, V, X, respectively.

[from Cay Horstmann: Java for Everyone, Wiley, 2013]

❯ Write a Dafny method that takes a positive integers as parameter and returns its representation in the Roman number system. Your method needs to have a contract, which Dafny proves to be valid.