Lab 4

In this lab your task is threefold: - you are asked to write a program given a specification - you are asked to write the specifications for a given known algorithm - you are asked to write both the algorithm and the specification starting from an informal specification.

Make sure that you determine the correct specifications (preconditions, postconditions, and loop invariants) and that they are complete, that is, that you do not miss properties that are essential to be satisfied by your code.

Every algorithm that you write is going to be implemented and verified using Dafny.

Example

Here is an informal specification: **write a program that returns the maximum between two numbers**.

Solution:

The informal specification does not specify what kind of numbers the program has to handle. This happens often in informal specifications, so in such cases, depending on the context, we may ask for clarification. Another way is to design the program in a way that allows us to modify it in the future, if possible (for example, if we assume that they are integers, most likely the way we write our program will allow us to extend it to real numbers).

Here is the formal spec:

```
Input: a, b -- integers
Output: max -- integer, such that max >= a and max >= b and (max == a or max == b)
```

The formal spec captures several important aspects of what our program should do: *first*, it establishes the signature of our program: it takes two integers and returns one; *second*, it establishes that the returned value is greater than or equal to both inputs; *third*, it establishes that the returned value is one of the inputs. It is very important not to miss this part.

Now, we write the spec in Dafny:

```
method Max(a: int, b: int) returns (max: int)
    ensures max >= a && max >= b
    ensures max == a || max == b
{
    ...
}
```

Note that we have no preconditions about the inputs. The only constraint is their type, captured in the signature of the function. The next step is to fill in the code so that the specification can be proved in Dafny. One possible implementation is below:

```
method Max(a: int, b: int) returns (max: int)
    ensures max >= a && max >= b
```

```
    ensures max == a || max == b
{
    if a > b {
        max := a;
    } else {
        max := b;
    }
}
```

Another possibility is:

```
method Max(a: int, b: int) returns (max: int)
    ensures max >= a && max >= b
    ensures max == a || max == b
{
    if a > b {
        return a;
    }
    return b;
}
```

Dafny is capable of proving either variant.

Exercises

1. (0.5p) Write a method that returns the maximum of three integers with complete specifications. Make sure you add the right postconditions to specify what properties the maximum value should satisfy.

2. (0.5p) Write a method that returns both the minimum and maximum of two numbers with complete specifications.

3. (0.5p) Specify the postconditions for the absolute difference between two integers.

   ```
   method AbsDiff(a: int, b: int) returns (diff: int)
       // Add your ensures clauses here
   {
       if a >= b {
           diff := a - b;
       } else {
           diff := b - a;
       }
   }
   ```

4. (0.5p) Add preconditions, postconditions, and loop invariants to verify this array maximum algorithm.

   ```
   method ArrayMax(a: array<int>) returns (max: int)
       // Add your requires clause here
       // Add your ensures clauses here
   {
       max := a[0];
       var i := 1;

       while i < a.Length
           // Add your invariants here
       {
           if a[i] > max {
               max := a[i];
           }
           i := i + 1;
   ```

```
    }
}
```

5. (1p) Write a program for finding the minimum element in an array. Write the complete specifications of this program and verify them using Dafny.

6. (1p) Write method that returns the difference between maximum and minimum elements of an array. Write the complete specifications of this program and verify them using Dafny.
