Lab 3: Dafny

Tom

1 Install software

The pink words below are hyperlinks.

1. This lab uses dafny to formally verify the real program.

2. Microsoft research provides tutorials from which you need to learn how to use Dafny. You need to focus on reading the Introduction, Methods, Pre- and Postconditions, Quantifiers, Array and Predicates sections. Other parts have little to do with this lab.

3. You can enter make method* to check your code (if checking method1, enter make method1). This command will check your program by calling dafny's executable file dafny/dafny and passing in the file name parameter. That is, call ./dafny/dafny xxx.dfy (where xxx.dfy is the name of the dafny code file you wrote).

   If the code is written correctly, you will see

   Dafny program verifier finished with 1 verified , 0 errors
   Compiled assembly into method1.dll

   If the code is written incorrectly, you will see

Dafny program verifier finished with 0 verified , 1 errors

What you want to make sure is that the number of errors is 0, otherwise there is something wrong with your code.

2 Problems

There are three small questions in this lab. You need to write a precondition or predicate.

2.1 method1

```
method method1(x: int, y: int) returns (z: int)
// Add a precondition here.
    ensure z > 0
{
    if x < 0
        { return y; }
    else
        { return x; }
}
```

Write an appropriate precondition to ensure that the return value is a positive number.

2.2 method2

```
method method2(a: array<int>, v: int) returns (b: int)
// Add a precondition here.
{
    return a[v] / v;
}
```

Write appropriate preconditions so that the program can pass verification and ensure that errors such as array out-of-bounds and divisor by 0 will not occur during runtime.

2.3 method3

```
predicate notzero(a: array<int>)
    reads a
{
// Add a predicate here.
}

method method3(a : array<int>, n : int) returns (b : int) requires n == a.Length && notzero(a)

    ensures b == 0;
{
    var i := 0;
    while i < n
            invariant 0 <= i <= a.Length
            invariant n == a.Length
            invariant forall k :: 0 <= k < i ==> a[k] != 0
    {
```

```
        if a[i] == 0
        { return 1; }

        i := i + 1;
    }
    return 0;
}
```

Write a suitable predicate so that the return value of the program must be 0.

3 Submit

The three pieces of code are placed in files named method1.dfy, method2.dfy, and method3.dfy respectively. After writing Run make handin in the lab3 directory to generate lab3.zip, and then upload it to canvas.