Instructor: Fei He

TA: Jianhui Chen
Fengmin Zhu

Due date: 05/26/2019

Assignment 8

stuname
stuid

Instructions: Write your answers in the corresponding `hw8.tex` file, compile it to a pdf, and hand the pdf file with all your Dafny source code, in a `.zip` archive, to *Tsinghua Web Learning* by the due date. Be sure to add your **student ID** and **full name** in the `stuid` and `stuname` macros at the top of `hw8.tex`. Problems marked with "Optional" are NOT mandatory. To receive full credits, you do NOT need to solve them. However, you will obtain extra points if you correctly solve them.

Acamedic Honesty: Any kind of plagiarism is strictly prohibited in the full semester for this course. Students who are suspected to copy other's work and is confirmed through investigation will receive no credits (i.e, zero) for this assignment. If you asked other students for help, or your referred to any material that is not provided by us (e.g. websites, blogs, articles, papers, etc., both online and offline), please mention them in your assignment (e.g. writing an acknowledgment, adding a reference).

1 Loops, Again (30 pts)

In this problem, you will practice finding loop invariants again, but in Dafny. Please first go through the attached file `Inv.dfy`.

(1) (10 pts) Find invariant(s) for the loop in method `PowerSeries`.

(2) (20 pts) Your elementary math teachers should have taught you that, the *greatest common divisor* of two integers m, n can be computed as follows:

$$gcd(m,n) = \begin{cases} m, & m = n \\ gcd(m-n,n), & m > n \\ gcd(m,n-m), & m < n \end{cases}$$

However, if we simply translate the above definition into Dafny code, as shown in the function `gcd`, the compiler complaints – she says, "cannot prove termination"! Now, you are asked to: (a) write an alternative definition which makes Dafny happy; (b) find invariant(s) for the loop in method `GCD`.

2 Termination (30 pts)

Consider the IMP program below. Prove that it will always terminate by providing a ranking function, a loop invariant and precondition (if necessary), listing the basic paths, their corresponding verification conditions, and arguing for their correctness.

Hint: You don't need to do this in Dafny, but you may find it helpful to test out ranking functions and invariants against its verifier.

```
$X := 1;$
while $X > 0$ do
   if $N \leq 100$ then
      $N := N + 11;$
      $X := X + 1$
   else
      $N := N - 10;$
      $X := X - 1$
   fi
end
```

Solution

We can prove the termination of the loop with the ranking function:

$$\delta(X, N) := (10 * X - N + max(100, n), X)$$

where n is the value of N on the entry.
Firstly, we define the compare function of two pairs:

$$(a, b) \leq (a', b') := (a < a') \vee (a = a' \wedge b \leq b')$$

First basic path:

```
$\{(0, 0) \leq \delta(X, N)\}$
$\downarrow \delta(X, N)$
$c_0 :$ assume $X > 0$
$c_1 :$ assume $N \leq 100$
$c_2 :$ $N := N + 11;$
$c_3 :$ $X := X + 1;$
$\downarrow \delta(X, N)$
```

The verification condition is:

$$(0, 0) \leq \delta(X, N) \rightarrow \delta(x) wlp(c_0; c_1; c_2; c_3, \delta(X, N) < \delta(X', N'))[X, N/X', N']$$
$$\hookrightarrow (0, 0) \leq \delta(X, N) \wedge X > 0 \wedge N \leq 100$$
$$\rightarrow (10 * (X + 1) - (N + 11) + max(100, n), X + 1) \leq (10 * X - N + max(100, n), X)$$

This verification condition is valid since $10 * (X + 1) - (N + 11) + max(100, n) < 10 * X - N + max(100, n)$.
Second basic path:

$$\begin{aligned}
&\{(0,0) \leq \delta(X, N)\}\\
&\downarrow \delta(X, N)\\
&c_0 : \ \textsf{assume } X > 0\\
&c_1 : \ \textsf{assume } N \leq 100\\
&c_2 : \ N := N - 10;\\
&c_3 : \ X := X - 1;\\
&\downarrow \delta(X, N)
\end{aligned}$$

The verification condition is:

$$\begin{aligned}
&(0,0) \leq \delta(X, N) \rightarrow \delta(x) wlp(c_0; c_1; c_2; c_3, \delta(X, N) < \delta(X', N'))[X, N/X', N']\\
\hookrightarrow &(0,0) \leq \delta(X, N) \wedge X > 0 \wedge N \leq 100\\
&\rightarrow (10 * (X - 1) - (N - 10) + max(100, n), X - 1) \leq (10 * X - N + max(100, n), X)
\end{aligned}$$

The verification condition is valid since $10 * (X - 1) - (N - 10) + max(100, n) = 10 * X - N + max(100, n)$ while $X - 1 < X$.
Therefore, the termination of the program gets proved.

3 A Verified Heap (40 pts)

A binary max-heap is a nearly complete binary tree that can be implemented in an array. Each node of the tree corresponds to an element of the array, and we track the following members to specify the structure:

1. The array a, which stores the root of the tree at index 1 (**not 0**), and should satisfy:
$$|a| = |nodes| + 1$$
2. The current number of nodes in the heap is tracked in *size*, and should satisfy:
$$0 \leq size \leq |a| - 1$$
3. The maximum number of nodes that the heap can store in *capacity*, and should satisfy:
$$capacity + 1 = |a|$$

At a given index i, we can easily compute the parents and children of the corresponding node:

$$\begin{aligned} parent(i) &= \lfloor i/2 \rfloor \\ left(i) &= 2i \\ right(i) &= 2i + 1 \end{aligned}$$

Max heaps satisfy a property which says that a parent is larger than both of its children:

$$\forall i. 1 < i \leq size \rightarrow a[i] \leq a[parent(i)]$$

In this problem, you will implement a binary max-heap with the three operations listed below. You are given specifications for each operation, and should implement a correct max-heap that satisfies these specifications. In other words, even though the specifications don't capture every property that correct heap operations do, you are expected to provide an implementation that does. So, for example, the insertion operation should place the given value at the correct location in the array, and the extraction operation should remove the greatest element from the heap, and return its value.

A sketch file `Heap.dfy` is already provided. Please complete the implementation and let Dafny compiler verify your code. You shall not modify any given code except the assume assignment statements we have asked you to remove when you are done. You may use outside resources to find more information about the algorithms that are typically used to implement these operations, but you should write the annotations necessary to satisfy the specifications yourself.

4 Heap Sort (20 pts, Optional)

Use your verified max-heap (in the last problem) to implement a heap sort algorithm, which sorts an array of integers in descending order. Complete the method body in `Heap.dfy`.