CSCI 2600 — Principles of Software

Homework 2: Reasoning About Loops

Due: Tuesday, Feb. 13, 2024, 11:59:59 pm

Introduction

In this assignment you will prove correctness of loops using the techniques we discussed in class.

Submission Instructions

- Follow the directions in the version control handout for cloning your hw02 git repo.The URI that you will need to use to clone your personal repo for this homework would have the form of `https://submitty.cs.rpi.edu/git/s24/csci2600/hw02/RCSID` where `RCSID` is your RCS ID. Submit your answers in a single .PDF file named `hw2_answers.pdf` in the `answers/` directory of your repository.
- Submit your Dafny code as `problem1.dfy`, `problem2.dfy`, `problem3.dfy`, and `problem4.dfy` files in the `answers/` directory of your repository. Submit your answers in a single .PDF file named `hw2_answers.pdf` in the answers/ directory of your repository. **You MUST type up your answers. Handwritten solutions will not be accepted or graded, even if they are scanned into a PDF file.** We recommend using LaTeX. If you have never used LaTeX, take a look at this tutorial.
- Be sure to commit and push the files to Submitty. Follow the directions in the version control handout for adding and committing files.
- Important: You must click the **Grade My Repository** button for you answers to be graded. If you do not, they will not be graded and you will receive a zero for the homework.

Problems

All answers should be placed in `answers/hw2_answers.pdf` except for Dafny files which should be separate files with names as given in each problem description.

Problem 1 (8 pts.): Sum of natural numbers

Below we give, in Java syntax, the `sumn` method which computes the sum of the n natural numbers from 1 to n. For example, $sumn(0) == 0$, $sumn(1) == 1$, $sumn(2) == 3$, and $sumn(6) == 21$.

```
// Precondition: n >= 0
int sumn(int n) {
  int i = 0, t = 0;
  while (i < n) {
    i = i + 1;
    t = t + i;
  }
  return t;
}
// Postcondition: t = n * (n + 1) / 2
```

a) Find a suitable loop invariant. (1 pt.)

b) Show that the invariant holds before the loop (base case). (1 pt.)

c) Show by induction that if the invariant holds after k-th iteration, and execution takes a k+1-st iteration, the invariant still holds (inductive step). (2 pts.)

d) Show that the loop exit condition and the loop invariant imply the postcondition $t = n*(n+1)/2$. (1 pt.)

e) Find a suitable decrementing function. Show that the function is non-negative before loop starts, that it decreases at each iteration and that when it reaches 0 the loop is exited. (1 pts.)

f) Implement the sum of natural numbers in Dafny. (2 pts., autograded)

- **Comment out** method `Main` in your Dafny code before you submit your code on Submitty.
- Method that implements the sum of natural numbers must be named `sumn` and have the following header: `method sumn(n: int) returns (t: int)`
- Make sure to include the precondition and the postcondition, as well as your invariant and the decrementing function.
- Verify your code with Dafny before submitting.
- Submit your Dafny code as a file named `problem1.dfy` in the `answers/` folder.

Problem 2 (15 pts.): Loopy square root

Below we give, in Dafny syntax, the square root method which should be computing the square root of a number.

```
method loopysqrt(n:int) returns (root:int)
    requires n >= 0
    ensures root * root == n
    {
        root := 0;
        var a := n;
        while (a > 0)
            //decreases //FILL IN DECREMENTING FUNCTION HERE
            //invariant //FILL IN INVARIANT HERE
            {
                root := root + 1;
                a := a - (2 * root - 1);
            }
    }
```

a) Test this code by creating the `Main()` method and calling `loopysqrt()` with arguments like $4, 25, 49$, etc. to convince yourself that this algorithm appears to be working correctly. In your answer, describe your tests and the corresponding output. (1 pt.)

b) Yet, the code given above fails to verify with Dafny. One of the reasons for this is that it actually does have a bug. More specifically, this code may produce the result which does not comply with the specification. Write a test (or tests) that reveals the bug. In your answer, describe your test(s), the corresponding outputs, and the bug that you found. Also, indicate which part of the specification is violated. (1 pt.)

c) Now find and fix this bug. Note that there might be several different ways of fixing the bug. Use the method that you think would be the best. You are not allowed to change the header of `loopysqrt()` or add, remove, or change any specifications or annotations. Do not worry about Dafny verifying the code for now, just fix the bug and convince yourself that `loopysqrt()` is now correct. You are also required to keep the overall algorithm the same as in the original version of the code. In your answer, describe how you fixed the bug and show the output of the same tests you ran before after fixing the bug. (2 pts.)

d) Update the specification of `loopysqrt()` to match the way you fixed the bug. If you changed the postcondition, make sure that it is the strongest possible postcondition. In your answer, describe your changes and explain why they were necessary. (1 pt.)

e) Does your Dafny code verify now? Why or why not? If it doesn't verify, does it mean that your code still has bugs in it? (1 pt.)

f) If your Dafny code doesn't verify, uncomment `invariant` and/or `decreases` annotations and supply the actual invariant and/or decrementing function. Make sure your code now

verifies. In your answer, describe how you guessed the invariant and/or decrementing function. Explain why your code was failing Dafny verification earlier but does verify now, despite the fact that you have not made any changes to your actual code (annotations are not part of the code). (1 pt.)

g) Finally, remove the precondition from your Dafny code and make necessary changes to the remaining annotations and/or code ensure that code still verifies. You are not allowed to change the header of `loopysqrt()`. You are also required to keep the overall algorithm the same as in the previous versions of the code. As before, make sure that your postcondition is the strongest. Did removing the precondition make your code more difficult? What effect did removing the precondition have on the client of `loopysqrt()`? (2 pts.)

h) Use computational induction to prove by hand the total correctness of the final version of your Dafny code. (6 pts.)

- **Comment out** method `Main` in your Dafny code before you submit your code on Submitty.
- Verify your code with Dafny before submitting. Check that your postcondition is the strongest.
- Submit your Dafny code as a file named `problem2.dfy` in the `answers/` folder. You only need to submit the last version of your `loopysqrt()` (the one with no precondition).

Problem 3 (12 pts.): Array of differences

Below is the pseudocode for creating an array containing elements each of which is the difference between two adjacent elements of the input array:

```
Precondition: arr != null ∧ arr.Length > 0
int[] difference(int[] arr) {
  diffs ← new int[arr.Length - 1]
  a ← 0
  while (a < diffs.Length)}
  {
    diffs[a] ← arr[a + 1] - arr[a]
    a ← a + 1
  }
  return diffs
}
Postcondition: diffs.Length = arr.Length - 1 ∧
  for all k, s.t. 0 ≤ k < diffs.Length: diffs[k] = arr[k + 1] - arr[k]
```

a) Find a suitable loop invariant. (2 pts.)

b) Show that the invariant holds before the loop (base case). (1 pt.)

c) Show by induction that if the invariant holds after k-th iteration, and execution takes a k+1-st iteration, the invariant still holds (inductive step). (4 pts.)

d) Show that the loop exit condition and the loop invariant imply the postcondition $result = i1+i2$. (1 pt.)

e) Find a suitable decrementing function. Show that the function is not negative before loop starts, that it decreases at each iteration and that when it reaches 0 the loop is exited. (2 pts.)

f) Implement the array of differences in Dafny. (2 pts., autograded)

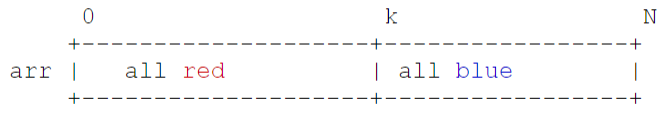
g) Extra credit (3 pts.) Implement `difference` in Dafny using sequences instead of arrays.

- **Do NOT** include method `Main` in your Dafny code that you submit on Submitty. If you use `Main` method for testing, make sure you comment it out before submitting on Submitty.
- Method that implements array of differences using arrays must be named `difference` and have the following header: `method difference(arr:array?<int>) returns (diffs:array<int>)`
- Function that implements array of differences using sequences instead of arrays must be named `difference` and have the following header: `function differences(arr:seq<int>):(diffs:seq<int>)`
- Make sure to include the precondition and the postcondition, as well as your invariant and the decrementing function.
- Verify your code with Dafny before submitting.
- Submit your Dafny code that uses arrays as a file named `problem3.dfy` in the `answers/` folder. If you decide to do the extra credit part with sequences instead of arrays, submit it as a file named `problem3s.dfy` in the `answers/` folder.

Problem 4 (15 pts.) The Simplified Dutch National Flag Problem

a) Given an array arr[0..N-1] where each of the elements can be classified as red or blue, write pseudocode to rearrange the elements of arr so that all occurrences of blue come after all occurrences of red and the variable k indicates the boundary between the regions. That is, all arr[0..k-1] elements will be red and elements arr[k..N-1] will be blue. You might need to define method swap(arr, i, j) which swaps the ith and jth elements of arr. (7 pts.)

The following picture illustrates the condition of the array at exit.



b) Write an expression for the postcondition. (2 pts.)

c) Write a suitable loop invariant for all loops in your pseudocode. (4 pts.)

d) Implement your pseudocode in Dafny. (2 pts., autograded)

- **Do NOT** include method `Main` in your Dafny code that you submit on Submitty. If you use `Main` method for testing, make sure you comment it out before submitting on Submitty.
- Represent each element of `arr` as either character 'r' (red) or character 'b' (blue).
- Method that solves the Simplified Dutch National Flag Problem must be named `dutch` and have the following header: `method dutch(arr: array?<char>) returns (k: int)` This method modifies `arr` and returns the value of `k`.
- Methods that modify `arr` might require a `modifies␣arr` annotation.
- You may find slicing and concatenating arrays useful. See Dafny Tutorial for more details and examples.
- In cases when your method changes the array, your conditions can refer to both new (current) and the old (previous) values by using the `old` keyword. For example, `arr[..] == old(arr[..])` means that arr has not changed.
- Make sure to include the precondition and the postcondition, as well as your invariant and the decrementing function.
- Verify your code with Dafny before submitting.
- Submit your Dafny code as a file named `problem4.dfy` in the `answers/` folder.

Collaboration (0.5 pts)

Please answer the following questions in a file named `collaboration.pdf` in your `answers/` directory.

The standard academic integrity policy applies to this homework.

State whether or not you collaborated with other students. If you did collaborate with other students, state their names and a brief description of how you collaborated.

Reflection (0.5 pts)

Please answer the following questions in a file named `reflection.pdf` in your `answers/` directory. Answer briefly, but in enough detail to help you improve your own practice via introspection and to enable me to improve Principles of Software in the future.

- In retrospect, what could you have done better to reduce the time you spent solving this homework?

- What could we, the teaching staff, have done better to improve your learning experience in this homework?
- What do you know now that you did not know before beginning the homework?

Submission

Push your repository containing the following files to Submitty:

- `answers/problem1.dfy`
- `answers/problem2.dfy`
- `answers/problem3.dfy`
- Optional `answers/problem3s.dfy`
- `answers/problem4.dfy`
- `answers/hw2_answers.pdf`
- `answers/collaboration.pdf`
- `answers/reflection.pdf`

Hints

- When trying to come up with a loop invariant for prewritten code, it often helps to trace through the execution of the code on paper. Choose a few different starting values of variables defined outside the block of code (such as method arguments), and write down the values of all the variables used in the loop for each iteration.
- Your Dafny code will be autograded by Submitty. If you are not getting full credit from the autograder, make sure to click "Show Details" in the autograding section to check the autograder output.

Errata

Check the Submitty Forum for possible updates or corrections.