CSC410

Assignment 2

Due on 2019/10/07, 2020 at 11:59pm

Assignment Format and Guidelines on Submission

This assignment is worth 10% of the total course mark. Submit on Markus and follow these rules:

- This is a group assignment, designed for groups of 4 students. The volume of work is designed so that it would be reasonable for a group of 4. You may choose to work in a group of 3. Groups of fewer than 3 students will be not be accepted. You cannot submit individually. You can use Piazza, Quercus, or the old-fashioned face-to-face to form groups.
- Each group should submit three files named `search.dfy`, `spy.dfy`, `stackinga.dfy`, `stackingb.dfy`, `mtsa.dfy`, and `mtsb.dfy`. Note that Markus has been setup to accept exactly those 3 files. All the required lemmas, helper functions, etc. should be included in each file for each problem.
- Each file should contain the code as given in the handout. Changing the algorithm will result in zero marks for that question unless otherwise specified.
- Assume statements and declarations without bodies will result in zero marks for that question. These basically admit facts without proofs into your reasoning and are disallowed for that reason.
- Method signatures for each method should remain exactly as specified in the handout. Changing the method signature to something incompatible with the original will result in zero marks for that question.

Note that your assignment will be automatically graded. Your function will be called from another function. If you mess with the signature, the call will fail and the autograder will give you a 0 mark.

The submission system will remain open for 12 hours after the deadline, but there is a penalty deduction formula set in Markus that deducts 4% for every hour of late submission up to 12 hours.

Word of Advice

Before you sit behind a Dafny terminal, make sure you have a detailed proof worked out on paper. If you don't have such a proof, you cannot hack your way through a Dafny proof. If you have such a proof, and are certain about its correctness, but cannot get it through to Dafny, then it most likely means that you are making a leap in reasoning somewhere that seems trivial to you, but not to the prover. We can assure you that this has nothing to do with a *feature* or a *command* that you do not know and have to dig up from a manual/tutorial. Everything you need to know to solve these has already been covered in class.

Problem 1 (30 points)

An implementation of a special case of a search routine with the precondition and postcondition encoded is given in the file `search.dfy` accompanying this assignment. Give a proof of correctness in Dafny for this search routine. Do not change the preconditions or postconditions. Only add annotations to the body of the method.

Problem 2 (40 points)

The accompanying Dafny code `mts.dfy` gives an implementation of the *maximum tail sum* problem. That is, given an array a of integers (both positive and negative values), it returns the sum of the maximum array interval beginning anywhere and ending at the end of the array. If all such sums are empty, it return 0 to reflect that an empty array interval (with a sum of 0) is optimal. This program is something that is known as a *programming pearl.* It is easy to implement this function in quadratic time, by enumerating all such intervals and computing their sums. But, this code is doing it in a single pass linear time. Your task is to prove that it does it correctly.

For grading purposes, we divide the full functional specification into two parts:

(a) (20 points) Prove that the first postcondition labeled as (a) in the file.

(b) (20 points) Prove the postcondition labeled as (b) in the file.

Problem 3 (40 points)

Consider the following game single player game. Initially, you are given a single stack of n boxes. In each turn of the game, you must perform exactly one of two moves:

1. Pick a stack of $a + b$ boxes and split it into two non-empty stacks of a and b boxes (i.e. $a, b > 0$). This move scores $a \times b$ points.
2. Pick a stack containing only one box and remove it. This actions scores no points.

The game is finished when no more moves are possible. No matter what moves you make, you will eventually run out of turns, and somewhat surprisingly, you will always finish the game scoring exactly $\frac{n(n-1)}{2}$ points. Try out a few runs of the game for some specific games to observe this.

Your task is to prove this outcome for the game. A Dafny encoding of the game is given in the accompanying file `stacking.dfy`. The program is simulating the game. Add the annotations necessary to prove to Dafny that the program (hence the game) terminates and that the postcondition holds. For grading purposes, we divide this assignment into the following two tasks:

Note: Dafny currently complains about line 14 of the code. This is intentional. You need to add a loop invariant that makes this complaint go away, and then be on your way about proving the problem correct.

(a) (20 points) Prove that the postcondition as specified in `stacking.dfy` and outlined above holds.

(b) (20 points) Prove that the loop terminates by providing the appropriate `decreases` clause for the loop and any argument that may be required to support it so that Dafny has no complaints about termination.

Problem 4 (30 points)

This problem is based on a known puzzle, "catch a spy".

A spy is located somewhere on a one-dimensional line, and he moves. He starts at location a. Every second, he moves b units to the right. Therefore, the location of the spy at time t is given by the expression $a + t * b$. But, the catch is that $a, b \in \mathbb{N}$ are constants that are unknown to us. We can only know check if the spy is at a particular location by querying that location, and we can only query one location at every second.

We can catch the spy as follows. The set $\mathbb{N} \times \mathbb{N}$ of pairs of natural numbers is countable [1]. Therefore, there exists a function $\mathit{unpair} : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ that covers all pairs of natural numbers.

Our strategy is to check the location $x + t \times y$ at time t, where $(x, y) = \mathit{unpair}(t)$. There exists some t_0 such that $(a, b) = \mathit{unpair}(t_0)$. This means that at time t_0 we will check the location $a + t_0 * b$, where the spy shall be at time t_0 as well! Therefore, we are guaranteed to catch the spy.

Now, you take the high level argument above and the sketch of an implementation that is given to you in file `spy.dfy` and turn it into a complete formal argument in Dafny. Complete the implementation of method *unpair* according to our strategy above, and prove that the while loop terminates (i.e. that we eventually catch the spy).

Hint: You should think about defining unpair recursively, instead of as a closed form function.

Note: The purpose is for us to practice using a theorem prover, like Dafny, to do a formal proof that is not related to a program. The puzzle and its solution are effectively provided to you (and you may consult resources online for this as well). We are not asking you to find a solution for the puzzle. We are only asking you to use Dafny to formally prove that the provided solution for the puzzle is indeed correct.

[1] If you do not know what countable means, then look it up.