Final exam (Tues, Dec 16, 1:30pm)

Goals of this course

- Write programs in the functional style
- Think recursively
- Understand sets, relations, and functions so that they can model real-world (and abstract) information
- Use formal logic to prove mathematical propositions.

Concepts of Chapter 7

- Sets and types can be defined recursively.
- Structural induction can be used to prove propositions quantified over recursively-defined set.
- Mathematical inductino can be used to prove propositions quantified over the whole (or natural) numbers.
- Loop invariants are used to capture the state of a computation that remains the same over all iterations of a loop while other aspects of the state change.
- Mathematical induction can be used to prove that a proposition quantified over the number of iterations is a loop invariant.

Concepts

7.1. Recursively-defined sets are specified using a subset of simple elements (the base case) and a subset of elements defined by a recursive rule. Recursively-defined sets are modeled by recursively-defined types. Python's class construct is used for programmer-defined types, including self-referential types. Classes consist in data components (instance variables) and function components (methods).

7.2. Full binary trees are a recursively-defined set; a full binary tree is either a node by itself (a leaf) or a node (an internal node) together with two children, which are full binary trees.

Standards

(See Standard 19 below.)

Standard 19. Write Python functions that operate on recursive types.

Concepts

7.4. Structural induction is a proof technique for propositions quantified over recursively defined sets, such as full binary trees. Structural-induction proofs entail a base case (for leaves) and an inductive case.

7.5. Mathematical induction is a proof technique for propositions quantified over whole numbers (or natural numbers). Like structural induction, proofs of mathematical induction entail a base case (for zero) and an inductive case.

Standards

Standard 20. Write proofs for propositions about recursively-defined sets using structural induction

(See standard 21 below)

Concepts

7.6. Imperative programming is characterized by statements, which are programming constructs that result in a side effect, such as modifying the value of variables. Loops are made up of a guard and a body; each execution of the body is an iteration.

7.7. Loop invariants are claims about the relationships among variables that remain true for all iterations.

Standards

Standard 21. Write proofs for loop invariants

7.4.2. For any full binary tree T, $\texttt{nodes}(T)$ is odd.

7.7.2. Prove the predicate to be a loop invariant for the loop in the given program.

$$I(n) = \text{ after } n \text{ iterations, } x + y = 100$$

```
def bbb(m) :
    x = 50
    y = 50
    i = 0
    while i < m :
        x += 1
        y -= 1
        i += 1
    return x + y
```