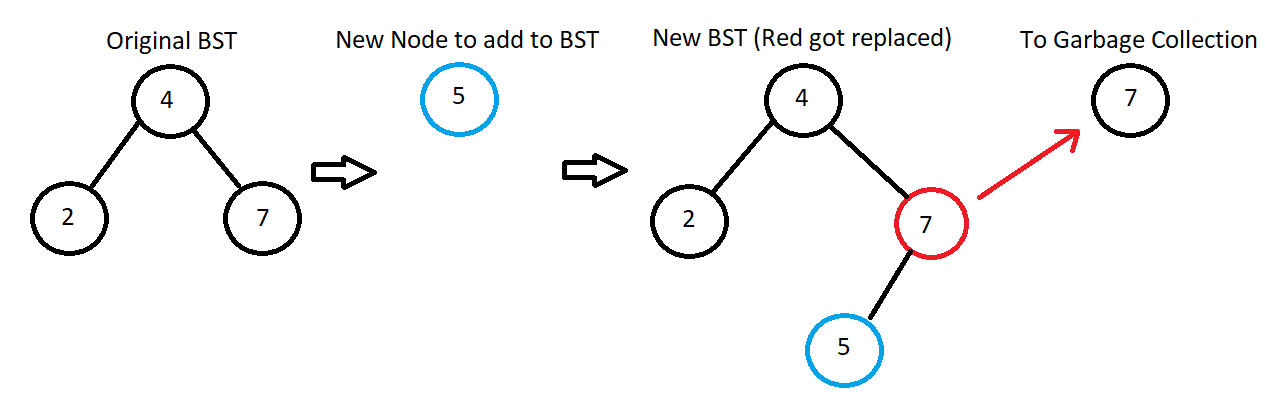
# CSE 382 – Patterns in Functional Programming Final Exam

By David Doria

**Explain the principle of persistence that exists with data structures written in a language that enforces immutability.  In addition to your written response, include graphics that show how persistence applies to a specific data structure of your choice.**

Persistence in a language that enforces immutability is used to prevent side effects in the code by preventing data from changing. Instead, when a change is required, these languages return completely new instances of the data with the modifications, and the old data goes out of scope and is collected by the garbage collector.

Here is a graphic of how persistence is applied to a Binary Search Tree. Unaffected nodes are reused, while nodes that would be mutated are replaced with a new node with the requested changes.



**Describe the benefits of writing specifications and definitions prior to implementing a function in code.**

Writing a specification and definition before implementing a function is a good way of organizing your thoughts before starting to implement that function. It allows you to have a map of sorts to follow while you implement it. It probably won’t be implemented precisely as it is written, but it will surely help.

**Provide a definition of what a functor is (from a programming perspective).  Identify the benefits of the functors we demonstrated in class over the traditional `for loops` found in languages like Python, C++, and Java.**

**Identify when it would be useful to curry a function.**

**Describe the process of modifying a function to support partial applications.**

**Identify when you should write a monoid function and explain its purpose.**

**Explain why the bind function is important if we have a monad type.**

**Describe the benefits of creating a stream function and explain how the function uses lazy evaluation.**

**Describe the purpose and the structure of a Min Heap.**

**Describe the purpose and the structure of a Random Access List.**