Exercise 1

• Implement and test a binary search tree
    • don’t worry about making it balanced
  ‣ implement key `insert()` and `lookup()` functions
    • bonus: implement a key `delete()` function
  ‣ implement it as a C module
    • `bst.c`, `bst.h`
  ‣ implement `test_bst.c`
    • contains `main()`, tests out your BST
Exercise 2

• Implement a Complex number module
  ‣ complex.c, complex.h
  ‣ includes a typedef to define a complex number
    • $a + bi$, where $a$ and $b$ are doubles
  ‣ includes functions to:
    • add, subtract, multiply, and divide complex numbers
  ‣ implement a test driver in test_complex.c
    • contains main()
Exercise 3

• Write a program that:
  ‣ prompts the user to input a string (use \texttt{fgets()}):
    • assume the string is a sequence of whitespace-separated integers
    • e.g., “5555 1234 4 5543”
  ‣ converts the string into an array of integers
  ‣ converts an array of integers into an array of strings
    • where each element of the string array is the binary representation of the associated integer
  ‣ prints out the array of strings
Exercise 4

• Modify the binary search tree code from exercise 1
  ‣ add static declarations to any internal functions you implemented in bst.h
  ‣ add a header guard to the header file
  ‣ write a Makefile
    • use Google to figure out how to add rules to the Makefile to produce a library (liblinkedlist.a) that contains the linked list code