CMSC 123: Data Structures

Handout Adapted from: Clinton Poserio, et al.

Exercise 04a: BST::Insert & BST::Search (Pre-Lab Exercise)

For this week, you begin implementing the `BST ADT` and two of its main operations - `insert` and `search`.

Tasks

For this exercise, the following functions are to be implemented:
1. `BST_NODE* createBSTNode(int)`
2. `BST* createBST()`
3. `void isEmpty(BST*)`
4. `void insert(BST*, BST_NODE*)`
5. `BST_NODE* search(BST*, int)`

These functions are already described in the handout and in `BST.h`.

Instructions

1. Implement the five (5) functions listed above.

2. Create a *test plan* for your implementation. A test plan is basically a list of operations to be executed to test that your implementation is correct.
   For example, here is a simple test plan:

   a. insert `x` into an empty BST.
   b. insert a value, `w` less than `x`.
   c. insert a value `y` greater than `x`. Take note of the correct or expected output of the operations. The operations above should produce a BST that looks like the following:

```
  y
x
  w
```

   It is also a good idea to create test plans for all possible cases of each operation.

3. Create a shell program for your test plan and store it to `test.cs`. Commands are described below. For example, the test plan given earlier could have the following:

```
+ 5
+ 3
+ 2
p
Q
```

4. Execute your test plan.

   a. Compile the interpreter program `main.c` together with your implementation `BST.c` (make sure you also have `BST.h`): `gcc -o run main.c BST.c`
   b. Execute `run` and use `test.cs` as input: `./run < test.cs`
   c. Compare your output with your expected output. If they are not the same, fix your code. Repeat testing until no more bugs/errors are found.
   d. If you have another test plan, *e.g.* stored in `test2.cs`, re-run the program with the new test plan: `./run < test2.cs`

Learn to test your code and *as much as possible*, avoid submitting code with compile errors *i.e.* code that don't even run.

Shell Program

A shell program is created to easily interact with the `BST` ADT. The available commands are described below:
+ `+ X` inserts the integer key `X` in the BST.
+ `? X` displays the location of the node with key `X`, if found.
+ `p` reports the contents of the BST in tree mode.
+ `Q` terminates the program.

Submission

Submit to our Lab Google Classroom a compressed (`.zip`) folder named prelab04a.zip (e.g.ajjacildoprelab04a.zip)containing the following:

1. `BST.h` and `BST.c`

2. `main.c`
3. `program.cs`

4. `Makefile`

Questions?

If you have any questions, contact your lab instructor.