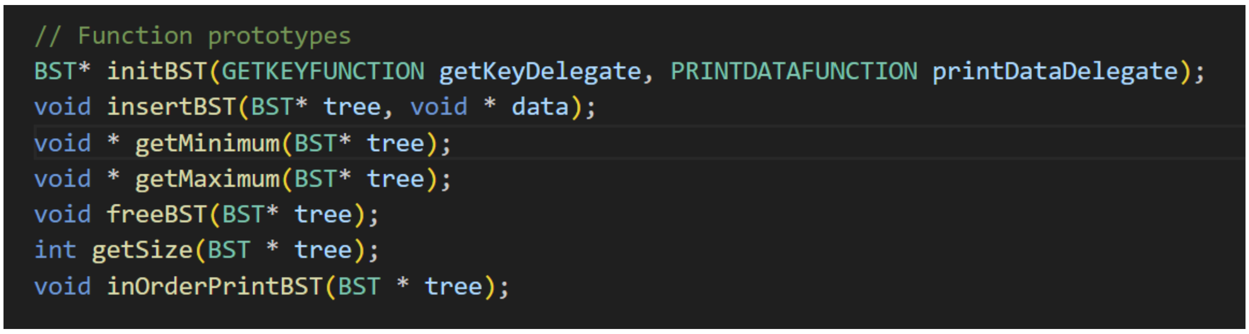
Lab #11
Spring 2025

Requirements
In this lab, you will make a BST library that is more general than in the past. This time, your BST should be able to take any data type. To do this, the user must provide a function to get the key from the data type and another function to print the data type. Take a look at the provided header file for prototypes and typedefs especially for the function pointer types you will use.

Notice that the header files contains "forward definitions" for the two structs used in the lab. These are just "place-holders" – they don't really tell you what the structures look like; only that such structures exist. It is up to you to actually define these structures in your implementation (lab11.c) file.

Hint: I would save off the function pointers somewhere so you can use them in your functions.

Functions
The functions for this lab are very similar to those you did last week. In fact, you can probably use your implementations from last week with a few changes. You can get the prototypes for the required functions from the supplied (lab11.h) header file, but here they are for your convenience:



```
// Function prototypes
BST* initBST(GETKEYFUNCTION getKeyDelegate, PRINTDATAFUNCTION printDataDelegate);
void insertBST(BST* tree, void * data);
void * getMinimum(BST* tree);
void * getMaximum(BST* tree);
void freeBST(BST* tree);
int getSize(BST * tree);
void inOrderPrintBST(BST * tree);
```

Hint: Our test code will provide an appropriate GETKEYFUNCTION and PRINTDATAFUNCTION for each data type we test. For your testing and debugging, it is highly recommended that you try your own data type (even if it is "int") with an appropriate GETKEYFUNCTION and PRINTDATAFUNCTION.

Rubric: 20 points
- 2 pts – initBST() correctly initializes and empty tree
- 2 pts – insertBST() correctly inserts nodes, including ALLOWING duplicates
- 2 pts – getMinimum() correctly returns the node with the smallest key
- 2 pts – getMaximum() correctly returns the node with the largest key
- 4 pts – freeBST() correctly deallocates memory (we will test with valgrind)
- 2 pts – getSize() correctly returns the number of nodes in the tree
- 6 pts – inOrderPrintBST() correctly prints all nodes in key order

Notice:
1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the −*Wall* and −*Werror* flags to be considered for a grade.

3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab policy document.