

## Overview

In this week's lab you will work with binary search trees, review and implement the fundamentals of the ADT, and use your constructed trees to observe the distinction between 'worst case' and 'expected' performance on real-world datasets.

## Pointers vs Reference vs Reference Pointers

Which of the following would change the value of input outside the function? Which functions *can* change the value if you change the contents (not the header) of the function?

```

pointers.cpp
void changeValue(int input){
    input = 10;
}

void changeValue(int * input){
    input = 10;
}

void changeValue(int & input){
    input = 10;
}

void changeValue(int *& input){
    input = 10;
}

```

What are the values of x and y after this program runs?

```

pointers.cpp
void changeValue(int *& input){
    *input = 1;
    input = nullptr;
}

```

```

int main(int argc, char** argv)
{
    int x = 42;

    int* y = &x;

    changeValue(y);
}

```

## Finding an element in a BST:

What are the return types for find() and \_find()?

```

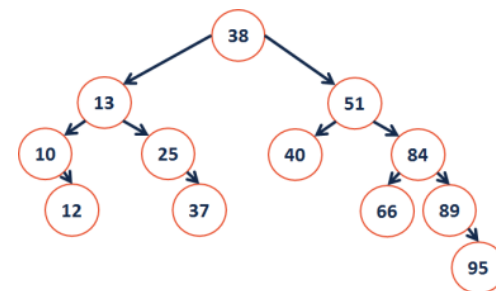
BST.hpp
template <typename K, typename V>
_____ find(const K & key) {
...
}

template <typename K, typename V>
_____ _find
(Node *& node, const K & key) {
}

```

What returns when we call:

Find(25):



Find(9):

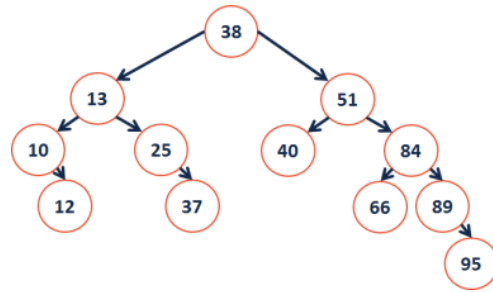
Running time? \_\_\_\_\_ Bound by? \_\_\_\_\_

**Inserting an element in a BST:**

Draw the changes to the tree when you insert:

**Insert(9)**

**Insert(81)**



Running time? \_\_\_\_\_ Bound by? \_\_\_\_\_

**Removing an element from a BST:**

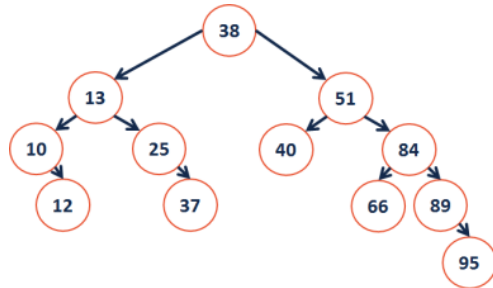
Redraw the tree after these changes.

`_remove (40)`

`_remove (25)`

`_remove (10)`

`_remove (13)`



Two Child Remove	

**The two child remove will always recurse to either a 0 or 1 child remove – why?**

Running time? \_\_\_\_\_ Bound by? \_\_\_\_\_

Zero-child Remove	One-child remove