MultiMaps and AVL Trees

Maps

In a normal map, the keys are unique and only one can exist. The relationship between a key and it's associated value is 1-to-1. For any given key there is only one value. A value can be repeated however. The map keeps the the values sorted by their id in a binary search tree data structure.

C++

```
struct Vehicle {
    int year;
    string make;
    string model;
    bool is_used;

    Vehicle() {
    }

    Vehicle(int new_year, string new_make, string new_model, bool new_is_used)
    : year(new_year), make(new_make), model(new_model), is_used(new_is_used) {
    }
};

int main() {
    // An example family car inventory
    map<int, Vehicle> car_inventory;

    // Insert some elements (Two different ways)
    car_inventory[1989] = Vehicle(1989, "Toyota", "Vehicleola", true);
    car_inventory.insert( { 2001, Vehicle(2001, "Ford", "Focus", false) } );

    cout << car_inventory[1989].make << endl;       // Prints out: Toyota
    cout << car_inventory[2001].model << endl;      // Prints out: Focus

    // The following will print out the key and the make like so:
    // 1989 Toyota
    // 2001 Ford
    for(auto elem : car_inventory) {
        cout << elem.first << " " << elem.second.make << endl;
    }
}
```

The key would be the year. The value would be a car object. The key in a map cannot be repeated.

Notice how in our for loop we need to iterate over each map element. The shortest way to do this is using the `auto` keyword. This will let the compiler figure out the type. This style of for loop will loop through every element in `car_inventory` and assign it to our `elem` named variable. The `auto` type is actually a `pair` object in this example since each map element stores both a key and a value. To access the key we want the first value in the pair element. If we want the value which is the Vehicle object, we need the second value in the pair.

Multimap

A multimap expands upon this and makes it so the keys are no longer unique. The relationship between a key and it's associated value is now many-to-many. So how does it keep track of multiple values with the same key? We can't sort values based solely on the keys now, we have to also sort based on the values themselves.

C++

```
int main() {
    // An example DMV database of cars
    multimap<string, Vehicle> dmv_car_ownership;

    // Insert some elements
    dmv_car_ownership.insert( { "Tham", Vehicle(1989, "Toyota", "Carola", true) } );
    dmv_car_ownership.insert( { "Tham", Vehicle(2001, "Ford", "Focus", false) } );
    dmv_car_ownership.insert( { "Trujillo", Vehicle(2004, "Toyota", "4Runner", true) }

    // The following will print:
    // Tham's Carola
    // Tham's Focus
    // Trujillo's 4Runner
    for (auto elem : dmv_car_ownership) {
        cout << elem.first << "'s " << elem.second.model << endl;
    }
}
```

The key this time would be the owner's last name. Notice how keys are not unique

Map Exercises

Answer the following questions and turn in your answers after the break. 1. Writing out code, declare a map with a key of type `char` and a value of type `string` .

1. What should the key, value pair be in a map for a list of items and their prices? Which is the key? Which is

the value? (The prices are not unique, the item names are)

2. Writing out code, add the item: Grapes for $4.95 to the map from question 2.
3. Write an example usage of multimap where names would not be unique (no code required). Describe why a multimap would be ideal over a normal map.
4. Write a for-loop to iterator over every element in a map, `cout` ing each key and their value.

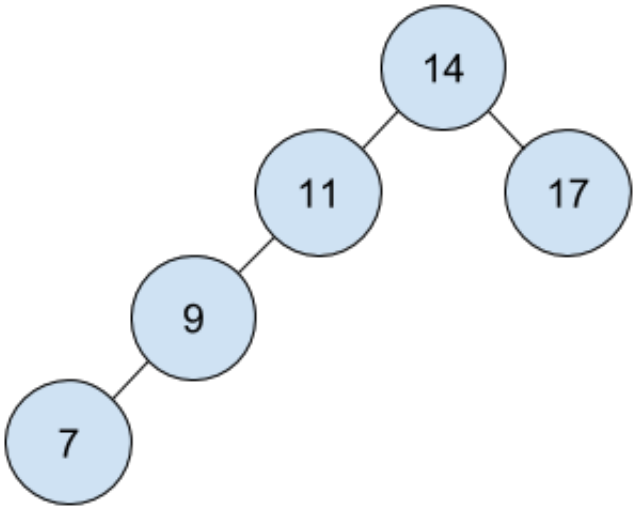
AVL Trees

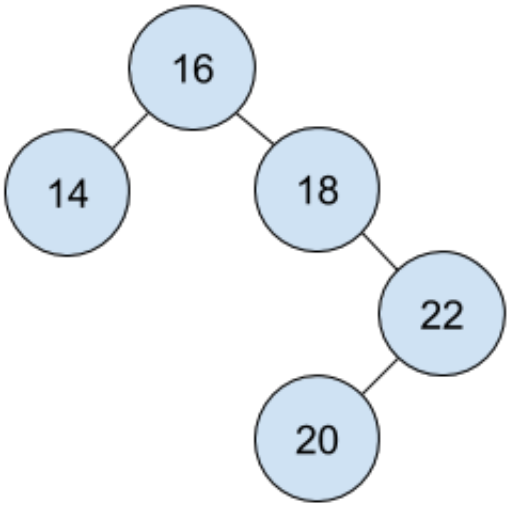
AVL Trees are self balancing trees who keep themselves balanced whenever the tree changes (either by the addition of a node or the removal). To achieve this, each node also needs to store the height. If the height difference between the left child and the right child differ by more than one, the tree needs to perform a rotation.

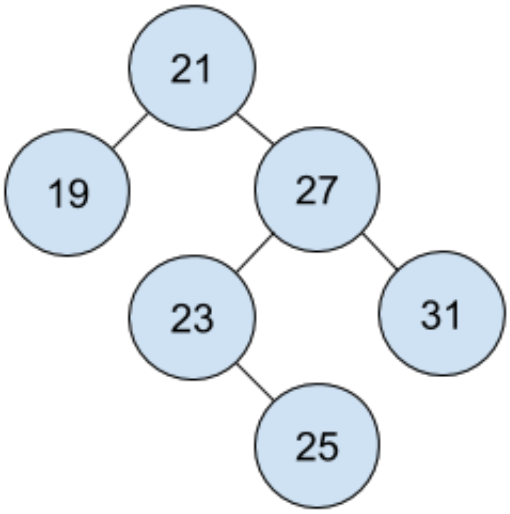
Using the worksheet: Located Here

AVL Exercises

Balance these structures so they are considered AVL trees. Draw the final result and write down which rotation you did.




16
14
18
22
20


21
19
27
23
31
25