# C++ B-Tree Implementation

## Overview

This program is an implementation of a B-Tree, a data structure that attempts to generalize a 2-3-4 tree by allowing between M/2 and M keys in a node.

The project implements three data structures: bTree, bTreeNode, and entry. However, the latter two are not meant to be used by third-parties, only by the bTree class as a means to store and operate on data.

## bTree

### bTree(int M)

```
bTree b = new bTree(M);
```

Creates a B-Tree that allows nodes to contain between M/2 and M keys. When M is an odd number, M/2 is rounded down.

### void insert(std::string key, std::string value)

```
b->insert("key", "value");
```

Inserts a key-value pair into the B-Tree.

### bool find(std::string key, std::string* value)

```
b->find("key", &value);
```

Attempts to find the element with the specified key. Returns true and sets value to the element's value when the item is found. Returns false when the item is not found.

### bool delete_key(std::string key)

```
b->delete_key("key");
```

Deletes the element with the specified key. Returns true when the deletion was successful and false when the key was not found.

### std::string toStr()

```
b->toStr();
```

Returns the contents of the B-Tree in string format. Items will be in order, delimited by the new line character ("\n").

## Known Bugs

- B-Tree has difficulty handling splits

## Tests

| Test case | Result |
|---|---|
| Odd M | Pass |
| Even M | Pass |
| Insert into empty tree | Pass |
| Delete from empty tree | Pass |
| Insert into full root | Fail |
| Delete item from almost underfilled node | Pass |
| Insert n keys, n > M | Fail |
| TC1 | Pass |
| TC2 | Fail |
| Find key in tree with n elements, n > M | Pass |
| Find missing key in tree with n elements, n > M | Pass |