HOMEWORK 3 -- Question 1

Instructions For Graders

- I'm using **Java**.
- My code has been tested with success on **crunchy5.cims.nyu.edu**.
- B-trees and Hash database are two separate programs.
- **Setup**: please put the newline-delimited file containing testing operations as **commands.txt** under **/input** folder.
- **Compile**:
  - B-trees: `javac BtreeTest.java`
  - Hash: `javac HashTest.java`
- **Run**:
  - B-trees: `java BtreeTest "../../../input/myindex" "../../../input/commands.txt"`
  - Hash: `java HashTest "../../../input/myindex" "../../../input/commands.txt"`
- **Output**: Stored under the **/output** folder as four files:
  - `btree_timing.txt` -- timings (individual and total) for btree experiment
  - `hash_timing.txt` -- timings (individual and total) for hash experiment
  - `btree_resultTable` -- query output and **final table** for btree experiment
  - `hash_resultTable` -- query output and **final table** for hash experiment

Folder Structure

- **/input** input path for data file and command file
- **/output** output path for timing experiments
- **/src/main/java** contains:
  - B-trees implementation, under **/btree**
  - Hash implementation, under **/hash**
  - Utility functions, under **/util**
    - credit to https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/
  - Helper class, that enables the storage of <Key, Value> pairs in a Java array, under **/pair**

Data Structure Implementation

BTree source:

- https://www.codeproject.com/Articles/1158559/B-Tree-Another-Implementation-By-Java

Hash source:

- https://github.com/phishman3579/java-algorithms-implementation/blob/master/src/com/jwetherell/algorithms/data_structures/HashMap.java

Compilation

BtreeTest class

```
javac BtreeTest.java
```

HashTest class

```
javac HashTest.java
```

Execution

BtreeTest class

```
java BtreeTest "../../../input/myindex" "../../../input/commands.txt"
```

HashTest class

```
java HashTest "../../../input/myindex" "../../../input/commands.txt"
```

Technical Specifications

Data & Index Storage

- Data is stored as a dynamically sized array (arraylist) of **<Key, Value>** pairs, where is treated as the natural order of the array.
- Hash and B-tree structures accept a and produce an into the data array.

Insert

- Implemented as **upsert**, meaning that if there's already a record with key value k, then that record's value is updated to v.
- **syntax**: `insert(<key>, <value>)`
- **output**:

- When inserting a record whose key DOES NOT already exist, the database output:

```
VALUE of KEY:<key> updated from <old_value> to <new_value>
Current number of items in <B-tree/HashMap> DB is: <number of entries>
```

- When inserting a record whose key already exist, the database output:

```
KEY:<key> VALUE:<value> inserted.
Current number of items in <B-tree/HashMap> DB is: <number of entries>
```

Delete

- **syntax**: `delete(<key>)`
- When a key is deleted, it is deleted from the B-tree/hash structure. But in order to **avoid shifting** the underlying data array, the Pair associated with the key stays in the array while its value is marked as **null**.
- **output**:
  - When deleting a record that exists in the database:

```
KEY: <key> deleted!
Current number of items in <B-tree/HashMap> DB is: <number of entries>
```

  - When deleting a record that DOES NOT exists in the database:

```
The KEY:<key> doesn't exist in the database.
```

Search

- **syntax**: `search(<key>)`
- **output**:
  - When searching for a key that exists in the database:

```
VALUE of the given KEY:<key> is: <value>
```

  - When deleting a record that DOES NOT exists in the database:

```
The KEY:<key> doesn't exist in the database.
```