1. Team member:

SJSU Students

2. Java file:
The top of your program will have those constant for SQLite local host

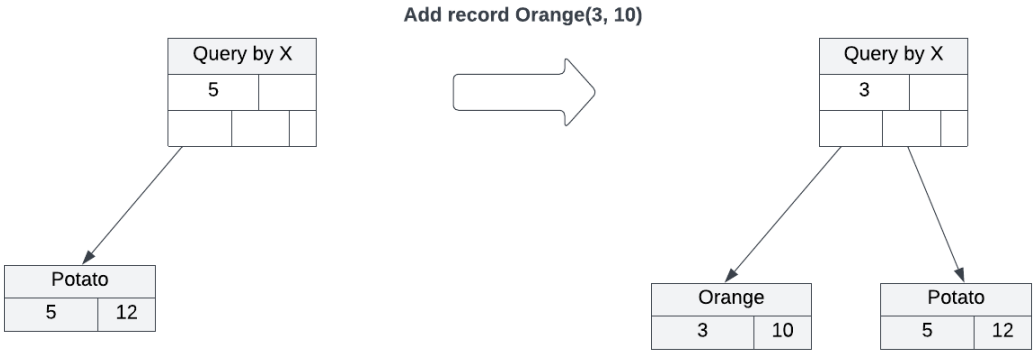
+ final static String SQLITE_URL = "jdbc:sqlite:";

3. Running SQLite instruction:
In case the local environment fails to work, please go to this link to get SQLite jar file and put it in hw1 folder https://search.maven.org/artifact/org.xerial/sqlite-jdbc
JDBC Version : 3.40.0.0
Compiler command: javac -cp sqlite-jdbc-3.40.0.0.jar:. TwoDimTreeManager.java
// In case the java fail to find related file in the package
Compiler command: javac TreePart.java TwoDimNode.java TwoDimRecord.java TwoDimTreeManager.java

Executing command: java -cp sqlite-jdbc-3.40.0.0.jar:. TwoDimTreeManager sqlite_database_name name_of_instruction_file

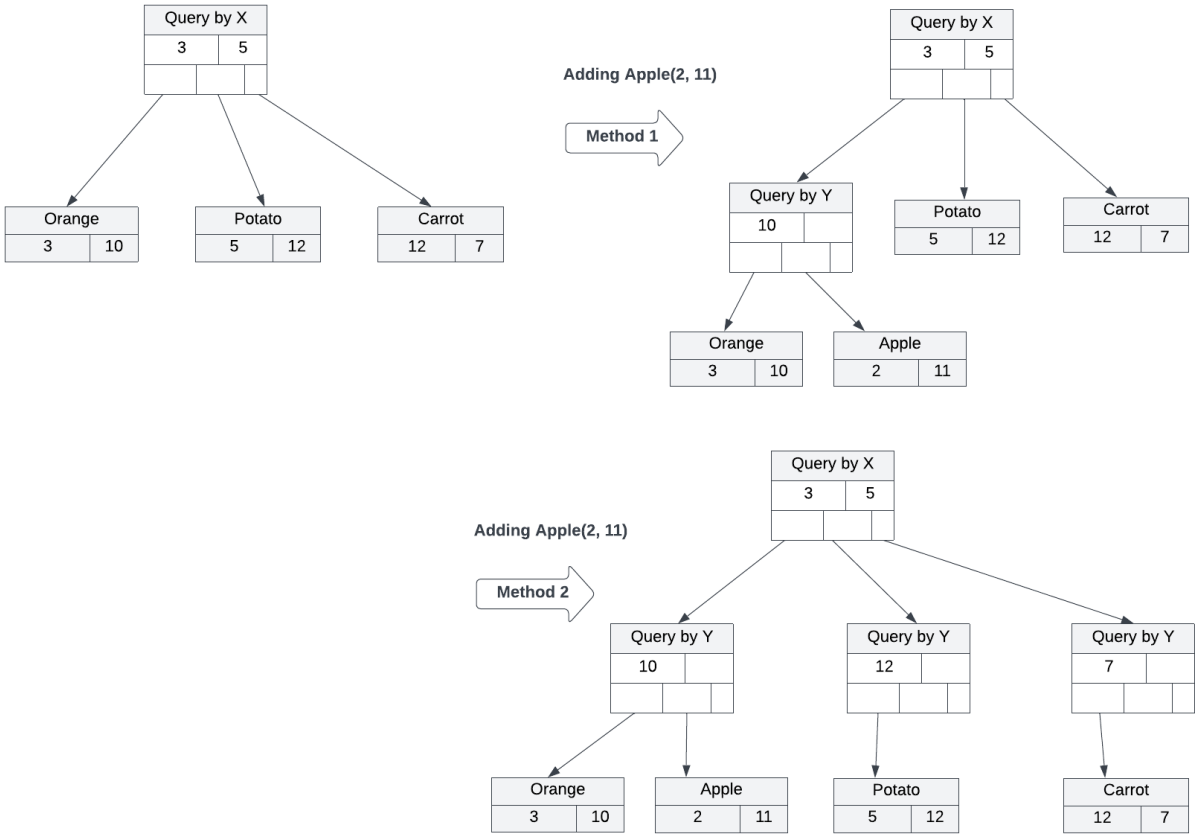
4. TwoDim Tree Logic:
I include this part since there might be multiple ways to implement the tree.

4.1 Inserting:
The key value of the inner node and its child position will adjust after inserting
Example:



4.2 splitting method:
There is two method of splitting:

+ 1st method we only create one innernode for the split children, then attach the children and new record to this node
+ 2nd method we create another tree level which then attach children to these node appropriately



	Method 1	**Method 2**
Pros	+Easy to implement +Save space due to have less inner node	+ Tree height is often sorter, so faster look up
Cons	+Higher tree compared to method 2, so look up less efficient	+Harder to implement due to more I/O and more leaf node rewiring to inner nodes. +More space for inner node

For my code: I chose method 1 for its simplicity

4.3 query key decision on each level:

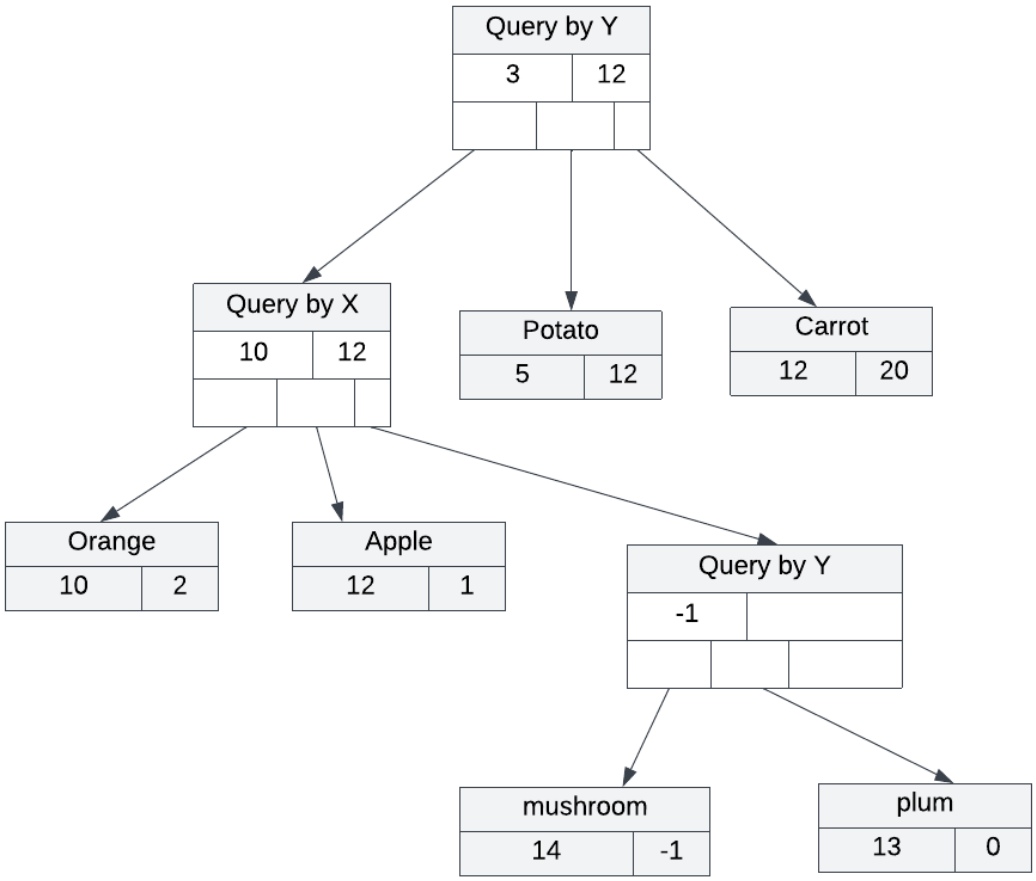
The root key will be default by Y key and it values will be:

+ value1 = smallest y of all children's y values

+ value2 = second smallest y of all children's y values

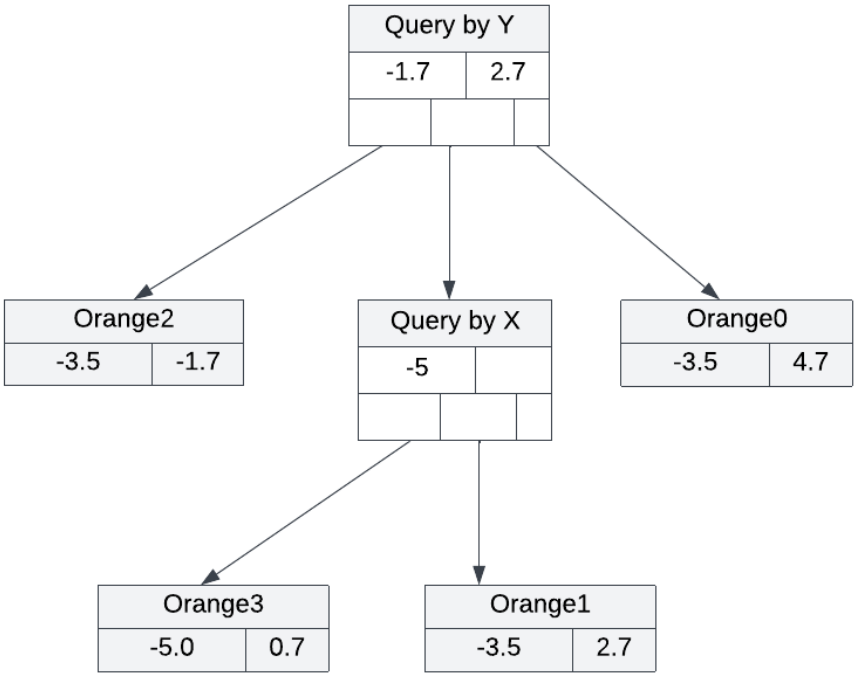
When splitting the key will alternate with the previous node key if we can split by that key, so that we can utilize and use all keys.

Example: the root key is Y, the next level is X, then the next level is query by Y. Also, the value of each level is follow the logic above



4.4 printTree:
Tree will be print in breadth first search order

For example: Orangetree below



Command: p OrangeTree
OrangeTree
Internal Node Id:7 Parent Id:-1 query variable is Y; key values are: -1.7, 2.7; Child IDs are: 10, 12, 8
Record: 10 Parent Id:7 Label:orange2 (-3.5, -1.7)
Internal Node Id:12 Parent Id:7 query variable is X; key values are: -5.0, NaN; Child IDs are: 11, 9, -1
Record: 8 Parent Id:7 Label:orange0 (-3.5, 4.7)
Record: 11 Parent Id:12 Label:orange3 (-5.0, 0.7)
Record: 9 Parent Id:12 Label:orange1 (-3.5, 2.7)