**Report**

1. Description
   1. Our B+-tree project contains 4 classes, which is Pair, Node, Tree and Test respectively.
   2. The Pair class represents the key value pair (key, rid) in B+-tree.
   3. The Node class represents the both inner nodes and leaf node in B+-tree and they are classified by a Boolean value **isLeafNode**. In our design, the node has several properties: **valueList** (the key values stored in each node), **pointerList** (the pointers of each node), **NODE\_FANOUT** (the maximum number of pointers in a node), **nextPointer** and **prePointer** (pointer to the next node and previous node), **changeLeafHeader** (). Additionally, the node provides methods: get and set methods for properties, **PrintNode** (print the key value inside a node), **Search** (get the satisfactory key values in the node and its subtree), **InsertValue** (insert the key or index value into suitable position of **valueList** of this node), **DeleteValue** (delete the key or index value into suitable position of **valueList** of this node), **Insert** (), **Delete** (delete the key value, return whether the value is deleted successfully or not and split the node or redistribute the key value), **Spilt** (spilt the inner or leaf node and return key value which needs to be added to upper node), **Merge** (merge the two node and rebuild the pointers), **BorrowLeaf** (for leaf node, borrow left or right siblings), **BorrowInner** (for inner node, borrow left or right siblings), **SolveUnderflow** (after deletion, try to borrow nodes from siblings, otherwise this method merges nodes of siblings)
   4. The Tree class represents the whole b+tree which has properties: **rootPointer** (root node pointer), **leafHeader** (the header of leaf nodes), **NODE\_FANOUT** (the number of fanout of a node). Our design provides methods like: **Search** (search the satisfactory results within the given the range and return a linked list contained the results), **DumpStatistics** (show the statistics of the b+-tree), **Insert** (search the right position and insert the value), **Delete** (search the desired position and delete value), **PrintTree** (print the whole tree as requirement)
2. Data Structure and Algorithm
   1. Linked List. We use the built-in linked list class of Java as the data structure to store the key values and pointers of every node.
3. Algorithm
   1. Recursion. We use recursion to realize the functions of search, insert, delete and so on so that the procedure can
   2. Simple Comparison. We use a simple for loop the find the expected key values or pointers.
4. Platform
   1. Running Platforms: Windows
   2. Programming Language: Java 8.0, sdk1.8
   3. Installation: Java 8.0
   4. Testing Procedure:
      1. With IDE (e.g. IntelliJ IDEA, eclipse)
         1. Choose the Test class
         2. Click run button
         3. Input the test command in the run window
         4. First type in btree DataFile.txt
         5. Then, use command like insert, delete, search, print, stat, quit
      2. With Command Line
         1. ???
5. Features
   1. User Interface
      1. We provide command line interactive user interface.
      2. We will check whether the input is valid or nor and give response.