Project 3 The B+ Tree Index Manager

In this experiment we are implementing an B+ tree. We intend to improve performance and efficiency when we access elements in the relation. By accomplishing the goal, we store <key, rid> pairs in the B+ Tree index. Buffer manager and efficient insertion of data records are important for the structure and performance.

We designed two helper functions to build the tree, createNode( ) and insertToNode( ).
The createNode function we allocate memory pages for each node and set their tree level to be 0. And then we insert them in to the tree. During the insertion we do the split check recursively. With these two functions we are able to create the tree. If duplicate keys were allowed, the simplest way to solve that is to use another disk memory to store the duplicated data. And add the address of the new memory block as a parameter of the former B+ tree. Data records are inserted one at a time. When the index file is created, the root of the B+ tree is initialized as a type of non-leaf root and page is unpinned as soon as it finishes. As we scan the next record, we make sure it satisfies the condition.

As for the testing, addition to the three tests provided, we add one more test specific for the situation when the relation contains negative values. Then test passed and when the data volume is very large, it may take a while for constructing a B+ tree.