Download the code that implements B-trees and extend it in the following ways:

- Modify the insertion operation to prevent duplicate elements from being inserted in the tree. If the user tries to insert an element that is already in the tree, your program should simply ignore the request.
- Write a method that returns the number of keys in the tree.
- Add a variable to the BTree object to store the number of keys in the tree and make sure the variable is kept up to date after every insertion operation (observe how the height of the tree is kept in a variable for efficiency, but it could also be easily computed).
- Write a method that returns the number of nodes in the tree.
- Add a variable to the BTree object to store the number of nodes in the tree and make sure the variable is kept up to date after every insertion operation.
- Determine if a given element is in the tree.
- Return the minimum element in the tree.
- Return the maximum element in the tree.
- Return the number of leaves in the tree.
- Return the number of nodes in the tree that are full.
- Extract all the elements in the tree into a sorted array.
- Print the contents of the tree ordered by depth, that is, first print the keys at depth 0, then the keys at depth 1, then the keys at depth 2, and so on. Hint: use a queue

As usual, write a report describing your work.