BTree Implementation
Fall 2016
(Due 11/17/2016, 11:59 pm)

Implement a BTree (**BT**). As you recall a BTree is a multi-way tree of degree ***m***. Where ***m*** defines the branching factor (or the maximum number of subtrees) for a node. For this assignment you can assume that the keys stored in the node are of type integers (your implementation must be generic, i.e. use templates). Your implementation you must provide the following methods:

- Insert key
- Find/Search for key
- Delete/Remove key
- Print tree (level order printing)

Key Functionality

- Command line arguments to load a driver file. No prompting or hard coding input data files. Your program will read commands from the driver file instead of from the user.
    - Commands in the driver file, one command per line
        - **L: loadfile**
          {open filename and insert keys into BTree}. Assume **keys are integer values** separated by a ***space. No assumption should be made about how many items are in the loadfile. Read until end of file.*** This will allow bulk-loading items in your BTree
        - **A: key**
          {adds the key to BTree }. If key is duplicate, print warning message, "**Warning, duplicate value, ignoring**"
        - **D: key**
          {removes **key** from BTree }. This will remove the specified value from BTree. Operation leaves your tree in a valid state, that is your tree satisfies conditions for BTree.
        - **S: key**
          {find/search for key in BTree}. Returns true of false
        - **P:**
          {prints BTree in level order}.
        - **T:**

{Terminate program}

You can generate your own data, please test your implementation.

Documentation. Please write a concise paragraph explaining your design philosophy and implementation. Each method you write should specify pre/post condition and type of arguments, and if arguments are modified.

Public Interfaces that will be tested

Interface	**Comment**	**Tree Height(1)**	**Tree Height(2)**	**Tree Height(>=3)**
insert(Type)	inserts object Type into tree{Debug message: if Type is a duplicate, print out message to that effect.}			
print()	print tree using level order			
find(Type)	return **true** if value of object Type is in tree, false otherwise			
delete(Type)	delete object with value of Type from tree {debug message: print if object was deleted or not found}			
reading from file	Pass command filename from command line.			
memory management	You must manage your memory allocation/deallocation. Failure to do so may result in a failing score			