Assignment - 4: Binary Search Tree on HardDisk

**Due Date:**  11.59 PM, April 22nd (Thursday Night)

**Total Marks:**  20 marks

**Problem statement**:

Implement a binary search tree on a harddisk.

Your tree should support the following operations (mentioned in the header file)

1. insert

2. delete based on the key

3. traversal - inorder : space separated values followed by \n

4. traversal - pre-order : space separated values followed by \n

As you guys know we can’t just store pointers directly in a file; they’d have no meaning even if you do write them. When it comes to a file, we represent addresses with file offsets. So, data structures like linked lists or trees will use offsets instead of pointers.

You may realize by now that you can’t just always have root at offset 0. We may call delete function which would invalidate that node and now your new root will be at some other offset.

Note that deletion of nodes creates holes in your file. You need to keep track of these invalidated nodes so that you can reuse the space when inserting new nodes, without unnecessarily increasing the size of the file.

So, the first structure in the file will be the header indicating the offset to the root node and offset to the linked list of free nodes. After the header, you can have your normal tree node with a data item, a left offset and a right offset. A value of -1 for offset can be used to indicate NULL.

free\_head is the offset to the head of a linked list that you would make using the BST nodes that have been deleted from the tree. To make a list with these nodes, just use either the left or the right offset to point to the offset of the next node.

In case of duplicate keys, just return from insert function. You should not add it again.

The header file is present in the drive folder. **Do not modify the header file.**

We will stop and restart the program to check whether the consistency is maintained and whether you’re storing the tree properly on the file or not.

**Please note that you’re not writing plaintext data to a file. This assignment is all about writing arbitrary binary data to a file.**

**Note:**

* Insertions & Deletions can be implemented iteratively or recursively
* If a node that has 2 children is deleted, replace it with its inorder successor
* Invalid offsets could be represented with -1
* The file pointed to by fp should be closed only in the close\_tree function. No other function should close the file.
* display\_preorder & display\_inorder should print to stdout
* If the tree is empty, \n should be printed for display\_preorder & display\_inorder
* Should not print anything for cases such as key not found, failed to open file, etc
* Ensure that implementations work fine on Ubuntu GCC compilers

**Files to be submitted:**

For this assignment you need to submit two files: implementation and readme.

A4\_<SRN>.c

A4\_README\_<SRN>.txt : Should contain the following:

* Key takeaway from this assignment.
* Additional notes

You can ask doubts at: [Doubt Clarifications](https://docs.google.com/spreadsheets/d/1pMnjK6HXHZYLRXddQPp3yp0hkXvw0saxsXYrKcmJNu0/edit?usp=sharing) (Use the assignment 4 sheet to ask your queries)

Submission link: <https://forms.gle/5XJtfdKgxdFQUYm17>

**Any cases of plagiarism will be dealt with seriously. DO NOT COPY/SHARE CODE. Do not take code from your peers or seniors or from online sources.**

**If you believe you can get away after plagiarising, think again.**