Write a C++ class implementing AVL tree considering the following class definition. You can add more functions (e.g., leftRotate, rightRotate) for your convenience.

```
class AVL_Node{
        private:
                int key;
                int bf;  // balance factor bf = height(left subtree) – height(right subtree)
                AVL_Node *LChild, *RChild;
        friend class AVL_Tree;
};

class AVL_Tree{
        private:
                AVL_Node *root;
        public:
                AVL_Tree();
                AVL_Tree(const AVL_Tree &T);
                AVL_Tree &operator=(const AVL_Tree &T);
                void AVL_Insert(int k);
                void AVL_Delete(int k);
                bool AVL_Search(int k);
                void AVL_Print(const char *filename);
                ~AVL_Tree();
};
```

1. **1.** [30 points] AVL_Insert(int k) – Inserts a node with key=k. If the element k already exists throw an exception.
2. **2.** [60 points] AVL_Delete(int k) – Delete the node with key = k. If no node exist throw an exception.
3. **3.** [5 points] AVL_serach(int k) – Finds the key k and returns true (false) if k is present (not present).
4. **4.** [10 points] AVL_Print(const char *filename) – Prints the tree structure with key and bf.
5. **5.** [10 points] Code readability (variable name, code organization, comment etc).
6. **6.** [30 points] Documentation explaining how you calculated balanced factor for each possible cases. Also document the output for sufficient test cases to show that you have covered all possible cases for maintaining the height balanced property.

Points to note:

1. Use gcc/g++ compiler and same code should run in both linux and windows (dev C++)

2. Your code should not leak any memory – use valgrind to check such issues.

3. Write a makefile (Windows user can install Cygwin)

4. You are not allowed to calculate height of the tree for adjusting balance factors during insertion and deletion.
5. Note that, we will use automated tool to check plagiarism in the source code and penalize heavily in case of unfair means and copying others' codes.

6. Maximum points possible = 150 points.

Submission Guidelines:

1. Submission deadline: Sept 9, 2021 11:59pm. Deadlines should be strictly followed. Late submissions will not be considered for evaluation.
2. Prepare a zip file containing source files (.cpp files), the PDF document and a readme file that clearly states how to run the code.
3. Name the zip file as BST_<rollno>.zip and send it to "cs513submission@gmail.com"