- *struct Node*
    - This will initialize every node in tree

- *class BKTree*
    - In this class we will define all function which are needed.

- *BKTree::BKTree()*
    - This is constructor where root=Null is defined

- BKTree::~BKTree()
    - This is deconstructor deleation operation on root is performed.

- *Node* BKTree::createNode(string w, size_t d)*
    - This function will create a node with left child and right child as node and store the distance between root and word

- *int BKTree::min(int a, int b, int c)*
    - The above function is used in levenshteinDistance function
    - This function find the minimum of three numbers

- sie_t BKTree::levenshteinDistance(string w1, string w2)
    - The above function is used in levenshteinDistance function two strings.
    - This function used the idea of dynamic programing
    - Dynamic folmula for this code is
    - dp[i][j]=min(dp[i-1][j]+1,dp[i][j-1]+1,dp[i-1][j-1]+1)

- void BKTree::recursiveSearch(Node* curNode, vector<string>& suggestions, string w, size_t t, bool& wordFound)

    - The above function is used to search the word in tree.w is word,curNode is root f the tree,t is 1,and wordFound is False.
    - If the word found in tree function will make wordfound=True

    - Else the function will store all the word whose difference from the word is -1 or +1 in suggestions

- void BKTree::add(string w)
    - The above fuction will add the string in tree.
    - If root is none it will define tree as None
    - Else if insert the node while find the proper parent of this node
- void BKTree::cleanString(basic_string<char>& s)
    - The above function will convert the string lowercase character
- void BKTree::search(string w, int t)
    - The above function will call search call the printSuggestions and recursiveSearch.
- void BKTree::printSuggestions(vector<string>& suggestions, bool wordFound)
    - The above function will tell if word is found in the tree or not
    - If word is found it will say word found
    - Else it will print the suggestions
- int main(int argc, const char * argv[])
    - this function will scan the words.file line by line create the tree
    - it will ask for the words to be searched

How to run the code?

- **compile the main.cpp file using g++ main.cpp**
- **Run the program using ./a.out**