***Problem Defination:-***

**“ A data set consisting of set of words is to be stored in the form of Tries.**

**Demonstrate create, insert and delete operations. ”**

***Detailed Description :-***

Trie is an efficient information **reTrieval** data structure. Using Trie, search complexities can be brought to optimal limit (key length). If we store keys in binary search tree, a well balanced BST will need time proportional to

**M \* log N**, where M is maximum string length and N is number of keys in tree.

Using Trie, we can search the key in O(M) time.

Maximum number of children of a node is equal to size of alphabet. Trie supports search, insert and delete operations in O(L) time where **L** is length of key.

***Real Time Applications :-***

* Autocompletion of strings.
* Dictionary

***Technology Stack : -***

The entire coding is done in Java with NetBeans IDE 8.0.1.

***Classes and Functions : -***

The program makes the use of 2 classes: -

1. **Trie** class.
   1. Trie\_ads\_assignment()
      * Constructor of the trie; i.e. creation of trie.
   2. private int getIndex(char ch)
      * this method returns the key index.
   3. public int search(String key)
      * this method uses the level order search for the key,such that if the first character of key doesn’t exist in the root,then simply return not found,else traverse the pointers level by level till the isLeafCharNode attribute of the node is true; and if so output the key and value printing key found.
   4. public void insert(String key, int value)
      * this method provides the insertion of the key with corresponding value as key-value pair,such that the key is taken character by character and each chararacter is assigned a new TrieNode as assigning the appropriate position in alphabetical order.
   5. private boolean hasNoChildren(TrieNode currentNode)
      * this method checks for the multiple paths for a node character such that it checks if it has no children ahead then it will return true.
   6. private boolean deleteHelper(String key, TrieNode currentNode, int length, int level)
      * this method provides level order searching for the character by character till the end of key;if found then a boolean variable deletedself will be set false; then top down approach such that level order deletion of nodes takes place and if a multiple path char node occurs it checks the further path node character and if it matches the char in key then delete it & further traverse till end,otherwise retain that unmatching pointer node character.
   7. public void delete(String key)
      * it checks for the empty trie or null key deletion,if not deleteHelper is called.
   8. public static void printTrie()
      * method display the current contents of trie.
   9. public static void main(String[] args)
2. **TrieNode** class.
   1. TrieNode(boolean isLeafNode, int value)
      * Node constructor
   2. public void markAsLeaf(int value)
      * marks the node as leaf,setting the isLeafCharNode = true;
   3. public void unMarkAsLeaf()
      * umarks the node from being a leaf, sets the isLeafCharNode = false;