Deccan Education Society's (DES)
Pune University, Pune
School of Engineering and Technology
Department of Computer Engineering and Technology
Program: B. Tech in Computer Science and Engineering

Academic Year: 2024-25	**Year:** Third Year	**Term:** II
PRN No.: 1012412079	**Name: Ratnajeet Patil**	
Subject: DS2		
Assignment No.: 2		
Date:		

Lab Assignment: 01

Title: 1. Create a Dictionary that stores keywords and its meanings, using appropriate data structure. Implement its operations such as add, delete, display, search and update its values. 2. Implement Binary Search Tree as Abstract Data Type and perform operations on it.

Code:

```java
import java.util.Scanner;

class Node {
    String keyword;
    String meaning;
    Node left, right;

    public Node(String keyword, String meaning) {
        this.keyword = keyword;
        this.meaning = meaning;
        this.left = this.right = null;
    }
}

class DictionaryBST {
    private Node root;

    public void add(String keyword, String meaning) {
        root = insert(root, keyword.toLowerCase(), meaning);
    }

    private Node insert(Node node, String keyword, String meaning) {
        if (node == null) return new Node(keyword, meaning);

        if (keyword.compareTo(node.keyword) < 0)
            node.left = insert(node.left, keyword, meaning);
        else if (keyword.compareTo(node.keyword) > 0)
            node.right = insert(node.right, keyword, meaning);
```

```
29.         else
30.             System.out.println("Keyword already exists. Use update instead.");
31.         return node;
32.     }
33.
34.     public String search(String keyword) {
35.         Node result = searchNode(root, keyword.toLowerCase());
36.         return (result != null) ? result.meaning : null;
37.     }
38.
39.     private Node searchNode(Node node, String keyword) {
40.         if (node == null || keyword.equals(node.keyword)) return node;
41.
42.         if (keyword.compareTo(node.keyword) < 0)
43.             return searchNode(node.left, keyword);
44.         else
45.             return searchNode(node.right, keyword);
46.     }
47.
48.     public void update(String keyword, String newMeaning) {
49.         Node node = searchNode(root, keyword.toLowerCase());
50.         if (node != null) {
51.             node.meaning = newMeaning;
52.             System.out.println("Meaning updated.");
53.         } else {
54.             System.out.println("Keyword not found.");
55.         }
56.     }
57.
58.     public void delete(String keyword) {
59.         root = deleteNode(root, keyword.toLowerCase());
60.     }
61.
62.     private Node deleteNode(Node node, String keyword) {
63.         if (node == null) return null;
64.
65.         if (keyword.compareTo(node.keyword) < 0)
66.             node.left = deleteNode(node.left, keyword);
67.         else if (keyword.compareTo(node.keyword) > 0)
68.             node.right = deleteNode(node.right, keyword);
69.         else {
70.             if (node.left == null) return node.right;
71.             if (node.right == null) return node.left;
72.
73.             Node min = findMin(node.right);
74.             node.keyword = min.keyword;
75.             node.meaning = min.meaning;
76.             node.right = deleteNode(node.right, min.keyword);
77.         }
78.         return node;
79.     }
80.
81.     private Node findMin(Node node) {
82.         while (node.left != null)
83.             node = node.left;
84.         return node;
85.     }
86.
87.     public void display() {
88.         if (root == null)
89.             System.out.println("Dictionary is empty.");
```

```
90.         else
91.             inOrder(root);
92.     }
93.
94.     private void inOrder(Node node) {
95.         if (node != null) {
96.             inOrder(node.left);
97.             System.out.println(node.keyword + " : " + node.meaning);
98.             inOrder(node.right);
99.         }
100.    }
101. }
102.
103. public class DictionaryBSTMain {
104.     public static void main(String[] args) {
105.         Scanner sc = new Scanner(System.in);
106.         DictionaryBST dict = new DictionaryBST();
107.
108.         while (true) {
109.             System.out.println("\n--- Dictionary using BST ---");
110.             System.out.println("1. Add");
111.             System.out.println("2. Delete");
112.             System.out.println("3. Search");
113.             System.out.println("4. Update");
114.             System.out.println("5. Display");
115.             System.out.println("6. Exit");
116.             System.out.print("Choose an option: ");
117.
118.             int choice = sc.nextInt();
119.             sc.nextLine();
120.
121.             switch (choice) {
122.                 case 1:
123.                     System.out.print("Enter keyword: ");
124.                     String keyword = sc.nextLine();
125.                     System.out.print("Enter meaning: ");
126.                     String meaning = sc.nextLine();
127.                     dict.add(keyword, meaning);
128.                     break;
129.                 case 2:
130.                     System.out.print("Enter keyword to delete: ");
131.                     String delKey = sc.nextLine();
132.                     dict.delete(delKey);
133.                     break;
134.                 case 3:
135.                     System.out.print("Enter keyword to search: ");
136.                     String searchKey = sc.nextLine();
137.                     String result = dict.search(searchKey);
138.                     if (result != null)
139.                         System.out.println("Meaning: " + result);
140.                     else
141.                         System.out.println("Keyword not found.");
142.                     break;
143.                 case 4:
144.                     System.out.print("Enter keyword to update: ");
145.                     String updateKey = sc.nextLine();
146.                     System.out.print("Enter new meaning: ");
147.                     String newMeaning = sc.nextLine();
148.                     dict.update(updateKey, newMeaning);
149.                     break;
150.                 case 5:
```

```
151.                     dict.display();
152.                     break;
153.                 case 6:
154.                     System.out.println("Exiting...");
155.                     sc.close();
156.                     return;
157.                 default:
158.                     System.out.println("Invalid choice.");
159.             }
160.         }
161.     }
162. }
163.
```

```
=== DICTIONARY MENU ===
1. Add word
2. Remove word
3. Look up meaning
4. Display all words
5. Count words
6. Exit
Enter choice (1-6): 1
Enter word: Hello
Enter meaning: Greeting
Word added successfully!

=== DICTIONARY MENU ===
1. Add word
2. Remove word
3. Look up meaning
4. Display all words
5. Count words
6. Exit
Enter choice (1-6): 3
Enter word to look up: Hello
Hello: Greeting

=== DICTIONARY MENU ===
1. Add word
2. Remove word
3. Look up meaning
4. Display all words
5. Count words
6. Exit
Enter choice (1-6): 4

===== DICTIONARY CONTENTS =====
Algorithm: A step-by-step procedure for solving a problem
Data Structure: A way of organizing data for efficient access and modification
Hello: Greeting
Java: A high-level, class-based programming language

=== DICTIONARY MENU ===
1. Add word
2. Remove word
3. Look up meaning
4. Display all words
5. Count words
6. Exit
Enter choice (1-6): 5
Total words: 4
```