AIWIR Lab Week 4

Team-8

Team members :

1.	A Spoorthi Alva	PES2UG19CS001
2.	A R Manyatha	PES2UG19CS002
3.	Achyut Jagini.	PES2UG19CS013
4.	Amulya S Dinesh	PES2UG19CS035

Problem statement : Create a dictionary using a hash table or BST. Compute the time complexities(W, B, A)

Tasks :

➔ Insert
➔ Delete
➔ Search
➔ Inorder traversal
➔ Preorder traversal
➔ Postorder traversal
➔ Time complexity

Language used : C

Libraries used :

➔ Time.h
➔ Stdio.h
➔ Stdlib.h
➔ String.h

We have used **BST** for dictionary implementation.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```
struct BSTnode {
      char word[128], meaning[256];
      struct BSTnode *left, *right;
};


struct BSTnode *root = NULL;

struct BSTnode * createNode(char *word, char *meaning) {
      struct BSTnode *newnode;
      newnode = (struct BSTnode *)malloc(sizeof(struct BSTnode));
      strcpy(newnode->word, word);
      strcpy(newnode->meaning, meaning);
      newnode->left = newnode->right = NULL;
      return newnode;
}

void insert(char *word, char *meaning) {
      struct BSTnode *parent = NULL, *current = NULL, *newnode = NULL;
      int res = 0;
      if (!root) {
              root = createNode(word, meaning);
              return;
      }
      for (current = root; current !=NULL;
         current = (res > 0) ? current->right : current->left) {
              res = strcasecmp(word, current->word);
              if (res == 0) {
                      printf("Duplicate entry!!\n");
                      return;
              }
              parent = current;
      }
      newnode = createNode(word, meaning);
      res > 0 ? (parent->right = newnode) : (parent->left = newnode);
      return;
```

```
}

void deleteNode(char *str) {
        struct BSTnode *parent = NULL, *current = NULL, *temp = NULL;
        int flag = 0, res = 0;
        if (!root) {
                printf("BST is not present!!\n");
                return;
        }
        current = root;
        while (1) {
                res = strcasecmp(current->word, str);
                if (res == 0)
                        break;
                flag = res;
                parent = current;
                current = (res > 0) ? current->left : current->right;
                if (current == NULL)
                        return;
        }
        /* deleting leaf node */
        if (current->right == NULL) {
                if (current == root && current->left == NULL) {
                        free(current);
                        root = NULL;
                        return;
                } else if (current == root) {
                        root = current->left;
                        free (current);
                        return;
                }

                flag > 0 ? (parent->left = current->left) :
                                (parent->right = current->left);
        } else {
                /* delete node with single child */
                temp = current->right;
                if (!temp->left) {
                        temp->left = current->left;
                        if (current == root) {
                                root = temp;
                                free(current);
                                return;
                        }
                        flag > 0 ? (parent->left = temp) :
                                        (parent->right = temp);
                } else {
                        /* delete node with two children */
                        struct BSTnode *successor = NULL;
                        while (1) {
                                successor = temp->left;
                                if (!successor->left)
                                        break;
                                temp = successor;
                        }
                        temp->left = successor->right;
                        successor->left = current->left;
                        successor->right = current->right;
                        if (current == root) {
                                root = successor;
                                free(current);
                                return;
                        }
                        (flag > 0) ? (parent->left = successor) :
                                        (parent->right = successor);
                }
        }
        free (current);
        return;
}

void findElement(char *str) {
        struct BSTnode *temp = NULL;
        int flag = 0, res = 0;
        if (root == NULL) {
                printf("Binary Search Tree is out of station!!\n");
```

```
               return;
       }
       temp = root;
       while (temp) {
               if ((res = strcasecmp(temp->word, str)) == 0) {
                       printf("Key : %s", str);
                       printf("Value: %s", temp->meaning);
                       flag = 1;
                       break;
               }
               temp = (res > 0) ? temp->left : temp->right;
       }
       if (!flag)
               printf("Search Element not found in Binary Search Tree\n");
       return;
 }

 void inorderTraversal(struct BSTnode *myNode) {
       if (myNode) {
               inorderTraversal(myNode->left);
               printf("Key    : %s", myNode->word);
               printf("Value: %s", myNode->meaning);
               printf("\n");
               inorderTraversal(myNode->right);
       }
       return;
 }


void preorderTraversal(struct BSTnode *myNode) {
       if (myNode) {

               printf("Key    : %s", myNode->word);
               printf("Value : %s", myNode->meaning);
               printf("\n");
               preorderTraversal(myNode->left);
               preorderTraversal(myNode->right);
       }
```

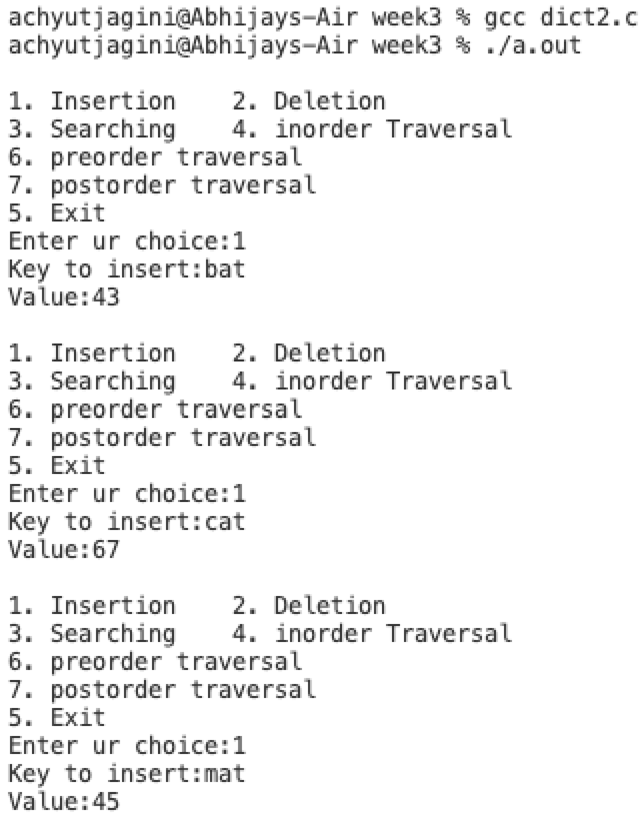
```c
    return;
}
void postorderTraversal(struct BSTnode *myNode) {
    if (myNode) {
        postorderTraversal(myNode->left);
        postorderTraversal(myNode->right);
        printf("Key    : %s", myNode->word);
        printf("Value : %s", myNode->meaning);
        printf("\n");

    }
    return;
}




int main() {
    int ch;
    char str[128], meaning[256];
    double time_spent = 0.0;
     clock_t end;
    clock_t begin = clock();

    while (1) {
        printf("\n1. Insertion\t2. Deletion\n");
        printf("3. Searching\t4. inorder Traversal\n");
        printf("6. preorder traversal\n");
        printf("7. postorder traversal\n");

        printf("5. Exit\nEnter ur choice:");

        scanf("%d", &ch);
        getchar();
        switch (ch) {
            case 1:
                printf("Key to insert:");
                fgets(str, 100, stdin);
```

```
                            printf("Value:");
                            fgets(meaning, 256, stdin);
                            insert(str, meaning);
                            break;
                    case 2:
                            printf("Enter the word to delete:");
                            fgets(str, 100, stdin);
                            deleteNode(str);
                            break;
                    case 3:
                            printf("Enter the search word:");
                            fgets(str, 100, stdin);
                            findElement(str);
                            break;
                    case 4:
                            inorderTraversal(root);
                            break;

                    case 6:
                        preorderTraversal(root);
                         break;

                     case 7:
                        postorderTraversal(root);
                         break;




                    case 5:
                        end = clock();
// calculate elapsed time by finding difference (end - begin) and
// dividing the difference by CLOCKS_PER_SEC to convert to seconds
       time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
       printf("The elapsed time is %f seconds", time_spent);
                       exit(0);

                    default:
```

```
                printf("You have entered wrong option\n");
                    break;
                 }
          }


     }
```

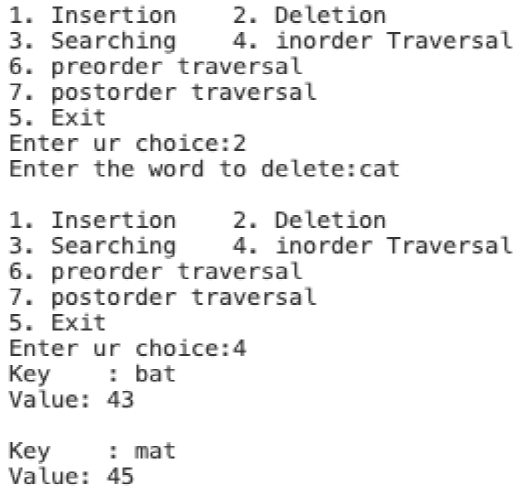
Output :

Case 1 :Insertion



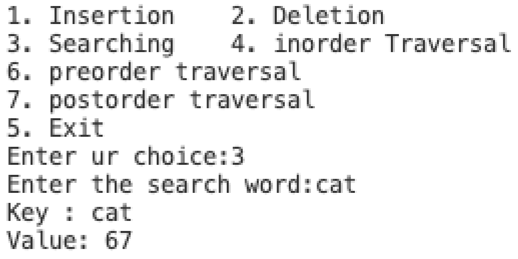
```
achyutjagini@Abhijays-Air week3 % gcc dict2.c
achyutjagini@Abhijays-Air week3 % ./a.out

1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:1
Key to insert:bat
Value:43

1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:1
Key to insert:cat
Value:67

1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:1
Key to insert:mat
Value:45
```

Case 2:Deletion



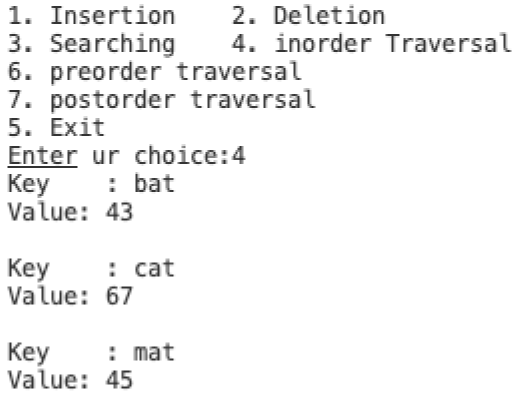
```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:2
Enter the word to delete:cat

1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:4
Key    : bat
Value: 43

Key    : mat
Value: 45
```

Case 3: Search



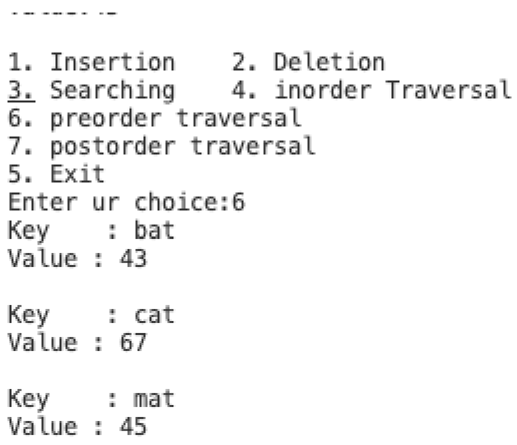
```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:3
Enter the search word:cat
Key : cat
Value: 67
```

Case 4: Inorder traversal



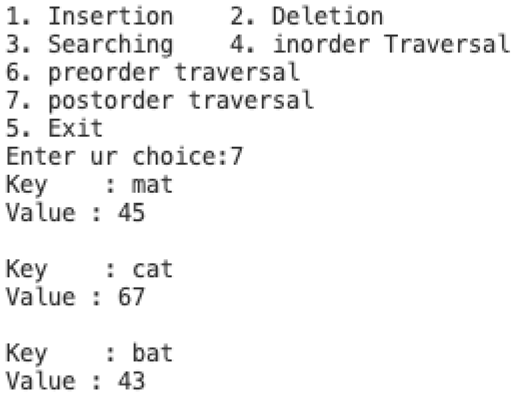
```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:4
Key    : bat
Value: 43

Key    : cat
Value: 67

Key    : mat
Value: 45
```

Case 6: Preorder traversal



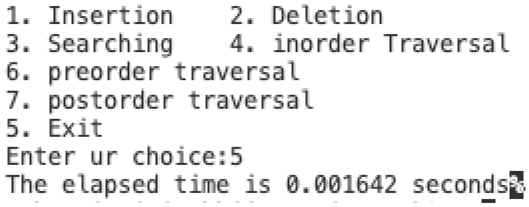
```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:6
Key    : bat
Value : 43

Key    : cat
Value : 67

Key    : mat
Value : 45
```

Case 7: Postorder traversal



```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:7
Key    : mat
Value : 45

Key    : cat
Value : 67

Key    : bat
Value : 43
```

Case 5: Exit (Along with elapsed time)



```
1. Insertion    2. Deletion
3. Searching    4. inorder Traversal
6. preorder traversal
7. postorder traversal
5. Exit
Enter ur choice:5
The elapsed time is 0.001642 seconds%
```