DSA Assignment: 9

Exp 9: Implementation of Binary Search Tree.

Shashwat Tripathi

D10A   Roll No: 60

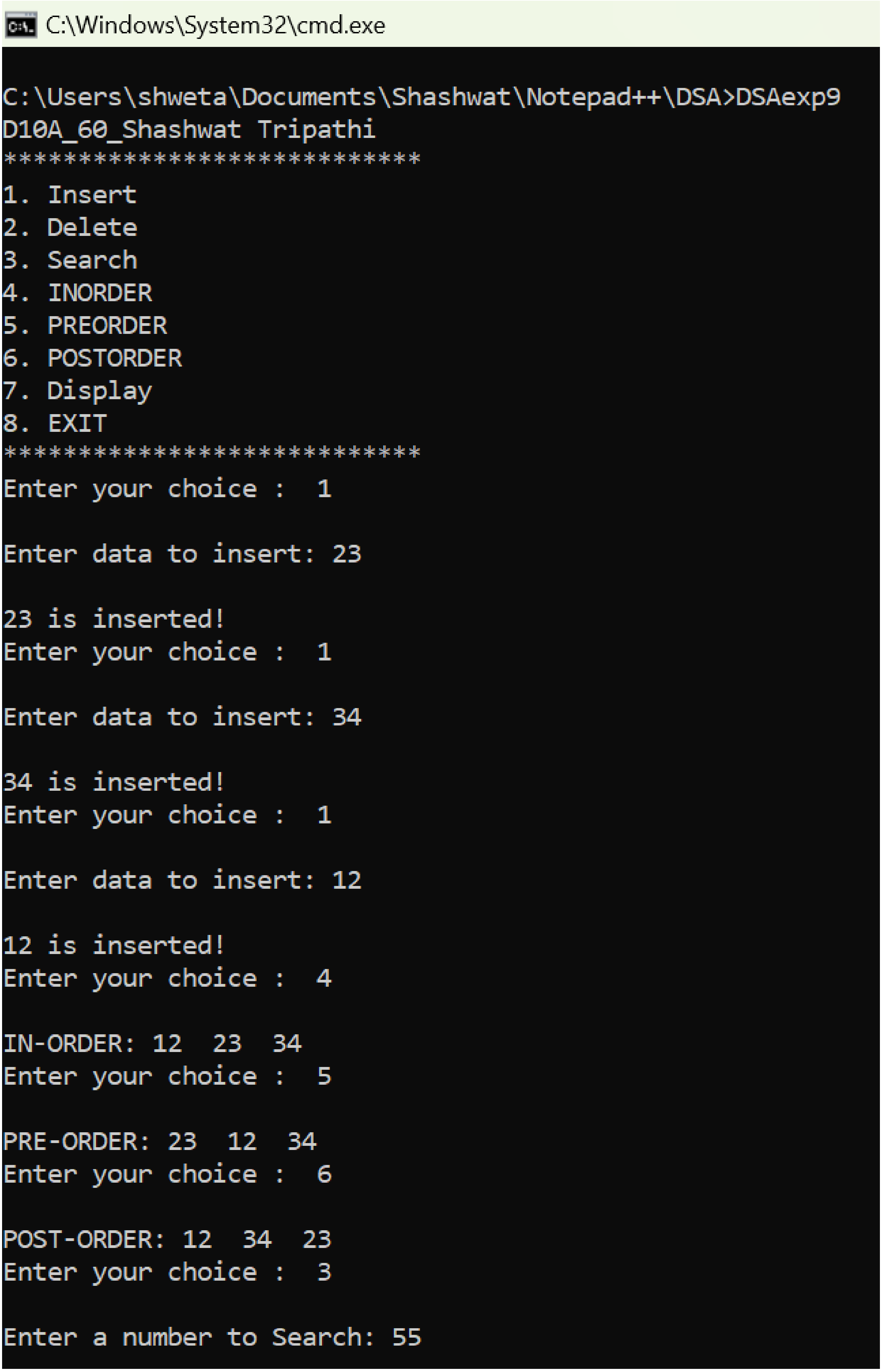
AIM: In this experiment, we will implement of Binary Search Tree.

CODE:

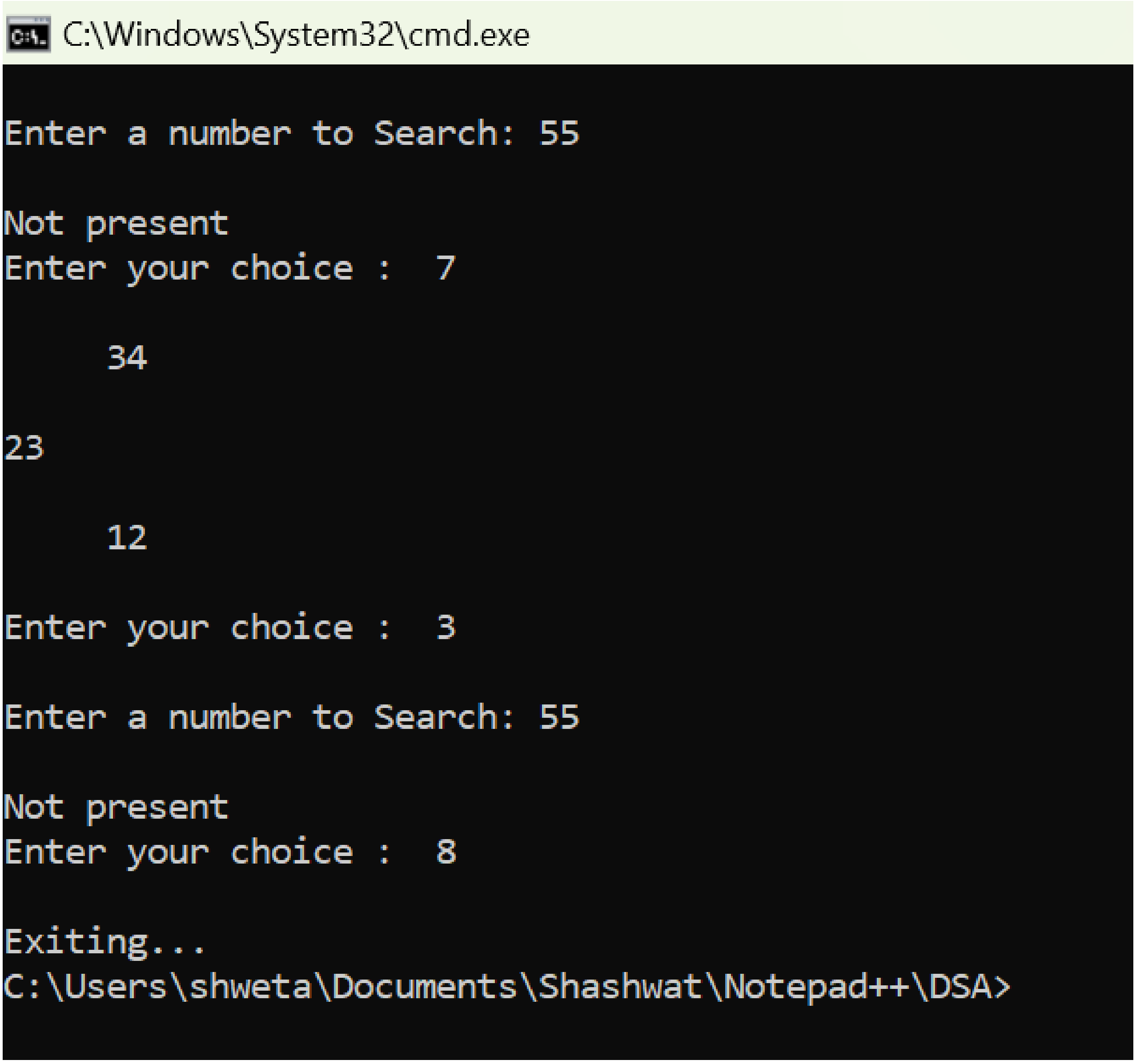
```
#include <stdio.h>
#include <stdlib.h>

struct node {
    struct node *left;
    int data;
    struct node *right;
};

struct node *root = NULL;

struct node *Min(struct node *root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}

struct node *getNewNode(int data) {
    struct node *newNode;
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data  = data;
    newNode->left  = NULL;
    newNode->right = NULL;
    return newNode;
}

struct node *insert(struct node *root, int data) {
    if (root == NULL) {
        root = getNewNode(data);
        return root;
    }
    if (data <= root->data) {
        root->left = insert(root->left, data);
    }
    else {
        root->right = insert(root->right, data);
    }

    return root;
}

struct node *delete(struct node *root, int val) {
    if (root == NULL) {
        return root;
    }
    else if (val < root->data) {
        root->left = delete (root->left, val);
    }
    else if (val > root->data) {
        root->right = delete (root->right, val);
    }
    else {
        if (root->right == NULL && root->left == NULL) {
            free(root);
            root = NULL;
        } else if (root->right == NULL) {
            struct node *temp = root;
            root = root->left;
            free(temp);
        } else if (root->left == NULL) {
            struct node *temp = root;
            root = root->right;
            free(temp);
        } else {
            struct node *temp = Min(root->right);
            root->data = temp->data;
            root->right = delete (root->right, temp->data);
        }
    }
    return root;
}

void search(struct node *root, int val) {
    if (root->data == val) {
        printf("\n%d is present in the tree", val);
        return;
    }
    if ((root->right == NULL && root->left == NULL) || root == NULL) {
        printf("\nNot present");
        return;
    }
    if (val <= root->data) {
        search(root->left, val);
    }
    else {
        search(root->right, val);
    }
}


void preOrderTraversal(struct node *root) {
    if (root == NULL) {
        return;
    }
    printf("%d  ", root->data);

    preOrderTraversal(root->left);

    preOrderTraversal(root->right);
```

```c
}

void inOrderTraversal(struct node *root) {
    if (root == NULL) {
        return;
    }

    inOrderTraversal(root->left);

    printf("%d  ", root->data);

    inOrderTraversal(root->right);

}

void postOrderTraversal(struct node *root) {
    if (root == NULL) {
        return;
    }

    postOrderTraversal(root->left);

    postOrderTraversal(root->right);

    printf("%d  ", root->data);
}

void printTree(struct node *root, int space) {
    if (root == NULL)
        return;

    space += 5;
    printTree(root->right, space);
    printf("\n");
    for (int i = 5; i < space; i++) {
        printf(" ");
    }
    printf("%d\n", root->data);
    printTree(root->left, space);
}

int main() {

    struct node *temp;
    int data, i, choice, val;
    printf("D10A_60_Shashwat Tripathi");
    printf("\n****************************");
    printf("\n1. Insert");
    printf("\n2. Delete");
    printf("\n3. Search");
    printf("\n4. INORDER");
    printf("\n5. PREORDER");
    printf("\n6. POSTORDER");
    printf("\n7. Display");
    printf("\n8. EXIT");
    printf("\n****************************");
  while (1) {
      printf("\nEnter your choice :  ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:printf("\nEnter data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                printf("\n%d is inserted!", data);
                break;

            case 2:printf("\nEnter a value to delete: ");
                scanf("%d", &val);
                root = delete (root, val);
                printf("\n%d is deleted!", val);
                break;

            case 3:printf("\nEnter a number to Search: ");
                scanf("%d", &data);
                search(root, data);
                break;

            case 4:printf("\nIN-ORDER: ");
                inOrderTraversal(root);
                break;

            case 5:printf("\nPRE-ORDER: ");
                preOrderTraversal(root);
                break;

            case 6:printf("\nPOST-ORDER: ");
                postOrderTraversal(root);
                break;

            case 7:printTree(root, 0);
                break;

            case 8:printf("\nExiting...");
                exit(1);
                break;

            default:printf("\nInvalid Choice..");
        }
    }
    return 0;
}
```

OUTPUT:



```
C:\Users\shweta\Documents\Shashwat\Notepad++\DSA>DSAexp9
D10A_60_Shashwat Tripathi
****************************
1. Insert
2. Delete
3. Search
4. INORDER
5. PREORDER
6. POSTORDER
7. Display
8. EXIT
****************************
Enter your choice :  1

Enter data to insert: 23

23 is inserted!
Enter your choice :  1

Enter data to insert: 34

34 is inserted!
Enter your choice :  1

Enter data to insert: 12

12 is inserted!
Enter your choice :  4

IN-ORDER: 12  23  34
Enter your choice :  5

PRE-ORDER: 23  12  34
Enter your choice :  6

POST-ORDER: 12  34  23
Enter your choice :  3

Enter a number to Search: 55
```



```

Enter a number to Search: 55

Not present
Enter your choice :  7


     34


23


     12


Enter your choice :  3

Enter a number to Search: 55

Not present
Enter your choice :  8

Exiting...
C:\Users\shweta\Documents\Shashwat\Notepad++\DSA>
```