

DSA Assignment

Boyapati Sai Venkat

AP19110010174

1st-year CSE-E.

Binary Search Tree

1. Write a menu-driven program to perform following operation on data structure

Binary Search Tree.

- a. Insert an element into the binary search tree.
- b. Inorder traversal (Display the content in inorder fashion)
- c. Preorder traversal (Display the content in preorder fashion)
- d. Postorder traversal (Display the content in postorder fashion)
- e. Searching an element in the binary search tree.
- f. Deletion of a given element from the binary search tree.

Solution:

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

struct node
{
    int data; //node will store an integer
    struct node *right_child; // right child
    struct node *left_child; // left child
}
```



```
};
```

```
void insert(struct node *root, int x);
```

```
void inorder();
```

```
void preorder();
```

```
void postorder();
```

```
void search(struct node *root, int x);
```

```
void delete(struct node *root, int x);
```

```
struct node* search(struct node *root, int x)
{
    if(root==NULL || root->data==x) //if root->data is x then the element is found
        return root;
    else if(x>root->data) // x is greater, so we will search the right subtree
        return search(root->right_child, x);
    else //x is smaller than the data, so we will search the left subtree
        return search(root->left_child,x);
}
```

```
//function to find the minimum value in a node
```

```
struct node* find_minimum(struct node *root)
```

```
{
```

```
if(root == NULL)
    return NULL;
else if(root->left_child != NULL) // node with minimum value will have no left child
    return find_minimum(root->left_child); // left most element will be minimum
return root;
}
```

//function to create a node

```
struct node* new_node(int x)
{
    struct node *p;
    p = malloc(sizeof(struct node));
    p->data = x;
    p->left_child = NULL;
    p->right_child = NULL;

    return p;
}
```

struct node* insert(struct node *root, int x)

```
{
    //searching for the place to insert
    if(root==NULL)
        return new_node(x);
    else if(x>root->data) // x is greater. Should be inserted to right
        root->right_child = insert(root->right_child, x);
    else // x is smaller should be inserted to left
```

```
    root->left_child = insert(root->left_child,x);
return root;
}
```

```
// function to delete a node
struct node* delete(struct node *root, int x)
{
    //searching for the item to be deleted
    if(root==NULL)
        return NULL;
    if (x>root->data)
        root->right_child = delete(root->right_child, x);
    else if(x<root->data)
        root->left_child = delete(root->left_child, x);

    // If item to be deleted in the leaf node
    else
    {
        //No Children
        if(root->left_child==NULL && root->right_child==NULL)
        {
            free(root);
            return NULL;
        }
    }
}
```

```
    }  
    // if item to be deleted has one child node  
    else if(root->left_child==NULL || root->right_child==NULL)  
    {  
        struct node *temp;  
        if(root->left_child==NULL)  
            temp = root->right_child;  
        else  
            temp = root->left_child;  
        free(root);  
        return temp;  
    }  
  
    //Two Children  
    else  
    {  
        struct node *temp = find_minimum(root->right_child);  
        root->data = temp->data;  
        root->right_child = delete(root->right_child, temp->data);  
    }  
}  
return root;  
}
```

```
void inorder(struct node *root)  
{  
    if(root!=NULL) // checking if the root is not null
```

```
{
    inorder(root->left_child); // visiting left child
    printf(" %d ", root->data); // printing data at root
    inorder(root->right_child); // visiting right child
}
}
```

```
void preorder(struct node* root)
```

```
{
    if(root == NULL) return;
    printf("%d ->", root->data);
    preorder(root->left_child);
    preorder(root->right_child);
}
```

```
void postorder(struct node* root)
```

```
{
    if(root == NULL) return;
    postorder(root->left_child);
    postorder(root->right_child);
    printf("%d ->", root->data);
}
```

```
void main()
```

```
{
    do{
        int input,x;
```

```

struct node *root = NULL;

printf("\n....menu driven c program.....");

printf("\n1.Inserting an elemnt into BST\n 2.Inorder traversal of BST\n 3.
Preorder traversal of BST\n 4. Post order traversl of BST\n 5.Deleting an elemnt into
BST\n 6.Searching an element in the BST\n 7.Exit\n ");

printf("\nEnter users choice");
scanf("%d", &input);
{
    if(input == 1)
    {
        printf("enter the element u want to insert");
        scanf("%d", &x);
        insert(x);
    }
    else if(input == 2)
    {
        struct node *root;
        printf("inorder trversal is :");
        inorder(x);
    }
    else if(input == 3)
    {
        struct node *root;
        printf("Preorder traversal is:");
        preorder(x);
    }
    else if(input == 4)
    {

```



```
    struct node *root;
    printf("Postorder traversal is: ");
    postorder(x);
}
else if(input == 5)
{
    struct node *root;
    printf("Enter the number to be deleted : ");
    scanf("%d",&x);
    del(x);

}
else if(input == 6)
{
    int number;
    printf("enter number to search");
    scanf("%d", &x);
    if(search == 1);
    {
        printf("Item found");

    }
    else
    {
        printf("item not found");
    }
}
```

```
        search();
    }
    else if(input == 7)
    {
        exit;
    }
    else
    {
        printf("Enter valid input");
    }

}while(0);

}
```

Output:

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted: 1

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted: 2

Item already present

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted : 3

Item already present

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted : 4

Item already present

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted: 5

Item already present

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted: 6

Item already present

....menu-driven c program.....


- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice1

Enter the number to be inserted: 7

Item already present

....menu-driven c program.....

- 
- 1.Inserting an element into BST
 - 2.Deleting an element into BST
 - 3.Inorder traversal of BST
 4. Preorder traversal of BST
 5. Postorder traversal of BST
 - 6.Searching an element in the BST
 - 7.Exit

Enter users choice2

Enter the number to be deleted: 7

Item not present in the tree

....menu-driven c program.....


- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice3

1

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST

- 
- 3.Inorder traversal of BST
 4. Preorder traversal of BST
 5. Postorder traversal of BST
 - 6.Searching an element in the BST
 - 7.Exit

Enter users choice4

1

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST
- 6.Searching an element in the BST
- 7.Exit

Enter users choice5

1

....menu-driven c program.....

- 1.Inserting an element into BST
- 2.Deleting an element into BST
- 3.Inorder traversal of BST
4. Preorder traversal of BST
5. Postorder traversal of BST



6.Searching an element in the BST

7.Exit

Enter users choice6

1

....menu-driven c program.....

1.Inserting an element into BST

2.Deleting an element into BST

3.Inorder traversal of BST

4. Preorder traversal of BST

5. Postorder traversal of BST

6.Searching an element in the BST

7.Exit

Enter users choice7