BINARY SEARCH TREE

```c
//Binary Search Tree Program

//header files
#include<stdio.h>
#include<stdlib.h>


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

//creating Node of Tree datastructure
struct Node* createNode(int data){
    struct Node* node;
    node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return node;
}

//inserting a new node in tree
struct Node* insert(struct Node* root, int data){
    // printf("lol1\n");
    if(root == NULL){
        root = createNode(data);
        // printf("lol3\n");
        }
    else{
        if(data < root->data)
            root->left = insert(root->left, data);
```

```
        else
            root->right = insert(root->right, data);
    }
    return root;
}

//left root right traversal
void printInorder(struct Node* node){
    if (node == NULL)
        return;
    printInorder(node->left); // first recur on left
child
    printf("%d ",node->data); // then print the data
of node
    printInorder(node->right); // now recur on right
child
}

//root left right traversal
void printPreorder(struct Node* node){
    if (node == NULL)
        return;
    printf("%d ",node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}

//left right root traversal
void printPostorder(struct Node* node){
    if (node == NULL)
        return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d ",node->data);
```

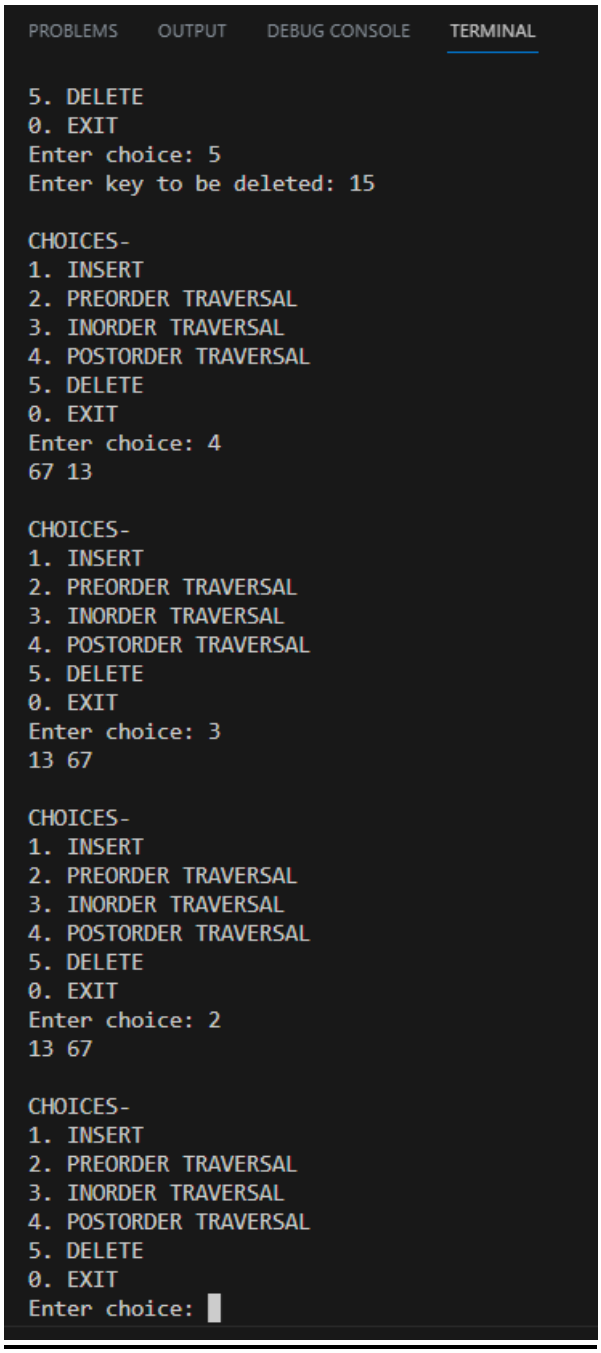
```
}

//binary search
struct Node* binarySearch(struct Node* root, int key){
    if(root == NULL)
        return NULL;
    if(root->data == key)
        return root;
    else if(key < root->data)
        return binarySearch(root->left, key);
    else
        return binarySearch(root->right, key);
}

//deleting a node in tree
struct Node *delete_node(struct Node* root, int data)
{
  if (root == NULL)
  {
    return root;
  }
  if (data < root->data)
  {
    root->left = delete_node(root->left, data);
  }
  else if (data > root->data)
  {
    root->right = delete_node(root->right, data);
  }
  else
  {
    if (root->left == NULL && root->right == NULL)
    {
      free(root);
```

```
      return NULL;
    }
    else if (root->left != NULL && root->right ==
NULL)
    {
      struct Node* temp;
      temp = root;
      root = root->left;
      free(temp);
      return root;
    }
    if (root->left == NULL && root->right != NULL)
    {
      struct Node *temp;
      temp = root;
      root = root->right;
      free(temp);
      return root;
    }
    else
    {
      struct Node* temp;
      struct Node* min;
      root = root->left;
      root = temp;
      while (root->right != NULL)
      {
        min = temp;
        temp = root->right;
      }
      root->data = min->data;
      root->left = delete_node(root->left, root-
>data);
      return root;
```

```
    }
    return root;
  }
}

int main()
{
    system("cls");
    struct Node *root = NULL;
    int key, choice;

    while(1)
    {
        printf("\nCHOICES-\n1. INSERT\n2. PREORDER
TRAVERSAL\n3. INORDER TRAVERSAL\n4. POSTORDER
TRAVERSAL\n5. DELETE\n0. EXIT\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter key to be added: ");
                scanf("%d", &key);
                root = insert(root, key);
                break;

            case 2:
                printPreorder(root);
                printf("\n");
                break;

            case 3:
                printInorder(root);
                printf("\n");
```

```
                break;

            case 4:
                printPostorder(root);
                printf("\n");
                break;

            case 5:
                printf("Enter key to be deleted: ");
                scanf("%d", &key);
                delete_node(root, key);
                break;

            case 0:
                exit(0);

            default:
                break;
        }
    }
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
5. DELETE
0. EXIT
Enter choice: 5
Enter key to be deleted: 15
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
0. EXIT
Enter choice: 4
67 13
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
0. EXIT
Enter choice: 3
13 67
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
0. EXIT
Enter choice: 2
13 67
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
0. EXIT
Enter choice: