

Q) program to find binary search tree, insertion, delete, search, traversal?

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

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

```
struct treeNode {
```

```
int data;
```

```
struct treeNode *left, *right;
```

```
};
```

```
struct treeNode *root = NULL;
```

```
struct treeNode* createNode(int data)
```

```
{
```

```
struct treeNode *newNode;
```

```
newNode = (struct treeNode *) malloc(sizeof (struct treeNode));
```

```
newNode->data = data;
```

```
newNode->left = NULL;
```

```
newNode->right = NULL;
```

```
return(newNode);
```

```
}
```

```
void insertion(struct treeNode **node, int data)
```

```
{
```

```
if (*node == NULL)
```

```
{
```

```
*node = createNode(data);
```

```
}
```

```

else if (data < (*node)->data)
{
insertion(&(*node)->left, data);
}
else if (data > (*node)->data)
{
insertion(&(*node)->right, data);
}
}

```

```

void deletion(struct treeNode **node, struct treeNode **parent, int data)
{
struct treeNode *tmpNode, *tmpParent;
if (*node == NULL)
return;
if ((*node)->data == data)
{
if (!(*node)->left && !(*node)->right)
{
if (parent)
{
if ((*parent)->left == *node)
(*parent)->left = NULL;
else
(*parent)->right = NULL;
}
}
}
}

```

```
free(*node);
}
else
{

free(*node);
}

}
else if (!(*node)->right && (*node)->left)
{
tmpNode = *node;
(*parent)->right = (*node)->left;
free(tmpNode);
*node = (*parent)->right;
}
else if ((*node)->right && !(*node)->left)
{
tmpNode = *node;
(*parent)->left = (*node)->right;
free(tmpNode);
(*node) = (*parent)->left;
}
else if (!(*node)->right->left)
{
```

```

tmpNode = *node;
(*node)->right->left = (*node)->left;
(*parent)->left = (*node)->right;
free(tmpNode);
*node = (*parent)->left;
}
else
{
tmpNode = (*node)->right;
while (tmpNode->left)
{
tmpParent = tmpNode;
tmpNode = tmpNode->left;
}
tmpParent->left = tmpNode->right;
tmpNode->left = (*node)->left;
tmpNode->right = (*node)->right;
free(*node);
*node = tmpNode;
}
}
else if (data < (*node)->data)
{
/* traverse towards left subtree */

```

```

deletion(&(*node)->left, node, data);
}
else if (data > (*node)->data)
{

deletion(&(*node)->right, node, data);
}
}
void findElement(struct treeNode *node, int data) {
if (!node)
return;
else if (data < node->data)
{
findElement(node->left, data);
}
else if (data > node->data)
{
findElement(node->right, data);
}
else
printf("data found: %d\n", node->data);
return;
}

void traverse(struct treeNode *node)

```

```
{  
if (node != NULL)  
{  
traverse(node->left);  
printf("%3d", node->data);  
traverse(node->right);  
}  
return;  
}
```

```
int main()  
{  
int data, ch;  
while (1)  
{  
printf("1. Insertion in BST\n");  
printf("2. Deletion in BST\n");  
printf("3. Search Element in BST\n");  
printf("4. Inorder traversal\n5. Exit\n");  
printf("Enter your choice:");  
scanf("%d", &ch);  
switch (ch)  
{  
case 1:  
while (1)
```

```
{  
    printf("Enter your data:");  
    scanf("%d", &data);  
    insertion(&root, data);  
    printf("Continue Insertion(0/1):");  
    scanf("%d", &ch);  
    if (!ch)  
        break;  
}  
break;  
case 2:  
    printf("Enter your data:");  
    scanf("%d", &data);  
    deletion(&root, NULL, data);  
    break;  
case 3:  
    printf("Enter value for data:");  
    scanf("%d", &data);  
    findElement(root, data);  
    break;  
case 4:  
    printf("Inorder Traversal:\n");  
    traverse(root);  
    printf("\n");  
    break;
```

```
case 5:  
    exit(0);  
default:  
    printf("you entered wrong option\n");  
    break;  
}  
}  
return 0;  
}
```



# Output

```
Terminal
1. Insertion in BST
2. Deletion in BST
3. Search Element in BST
4. Inorder traversal
5. Exit
Enter your choice:1
Enter your data:3
Continue Insertion(0/1):0
1. Insertion in BST
2. Deletion in BST
3. Search Element in BST
4. Inorder traversal
5. Exit
Enter your choice:2
Enter your data:3
1. Insertion in BST
2. Deletion in BST
3. Search Element in BST
4. Inorder traversal
5. Exit
Enter your choice:3
Enter value for data:5
1. Insertion in BST
2. Deletion in BST
3. Search Element in BST
4. Inorder traversal
5. Exit
Enter your choice:4
Inorder Traversal:
0
1. Insertion in BST
2. Deletion in BST
3. Search Element in BST
4. Inorder traversal
5. Exit
Enter your choice:5
5
```