```
#include <stdio.h>

#include <stdlib.h>

struct BST{

int data;

struct BST *left;

struct BST *right;

};

typedef struct BST NODE;

NODE *node;

NODE* createtree(NODE *node, int data)

{

if (node == NULL)

{

NODE *temp;

temp= (NODE*)malloc(sizeof(NODE));

temp->data = data;
```

```
temp->left = temp->right = NULL;

return temp;

}

if (data < (node->data))

{

node->left = createtree(node->left, data);

}

else if (data > node->data)

{

node -> right = createtree(node->right, data);

}

return node;

}

NODE* search(NODE *node, int data)

{

if(node == NULL)
```

```
printf("\nElement not found");

else if(data < node->data)

{

node->left=search(node->left, data);

}

else if(data > node->data)

{

node->right=search(node->right, data);

}

else

printf("\nElement found is: %d", node->data);

return node;

}

void inorder(NODE *node)

{

if(node != NULL)
```

```
{
inorder(node->left);
printf("%d\t", node->data);
inorder(node->right);
}
}
void preorder(NODE *node)
{
if(node != NULL)
{
printf("%d\t", node->data);
preorder(node->left);
preorder(node->right);
}
}
void postorder(NODE *node)
```

```
{
if(node != NULL)
{
postorder(node->left);
postorder(node->right);
printf("%d\t", node->data);
}
}
NODE* findMin(NODE *node)
{
if(node==NULL)
{
return NULL;
}
if(node->left)
return findMin(node->left);
```

```
else

return node;

}

NODE* del(NODE *node, int data)

{

NODE *temp;

if(node == NULL)

{

printf("\nElement not found");

}

else if(data < node->data)

{

node->left = del(node->left, data);

}

else if(data > node->data)

{
```

```
node->right = del(node->right, data);

}

else

{ /* Now We can delete this node and replace with either minimum element in the right sub tree or maximum element in the left subtree */

if(node->right && node->left)

{ /* Here we will replace with minimum element in the right sub tree */

temp = findMin(node->right);

node -> data = temp->data;

/* As we replaced it with some other node, we have to delete that node */

node -> right = del(node->right,temp->data);

}

else

{

/* If there is only one or zero children then we can directly remove it from the tree and connect its parent to its child */

temp = node;

if(node->left == NULL)
```

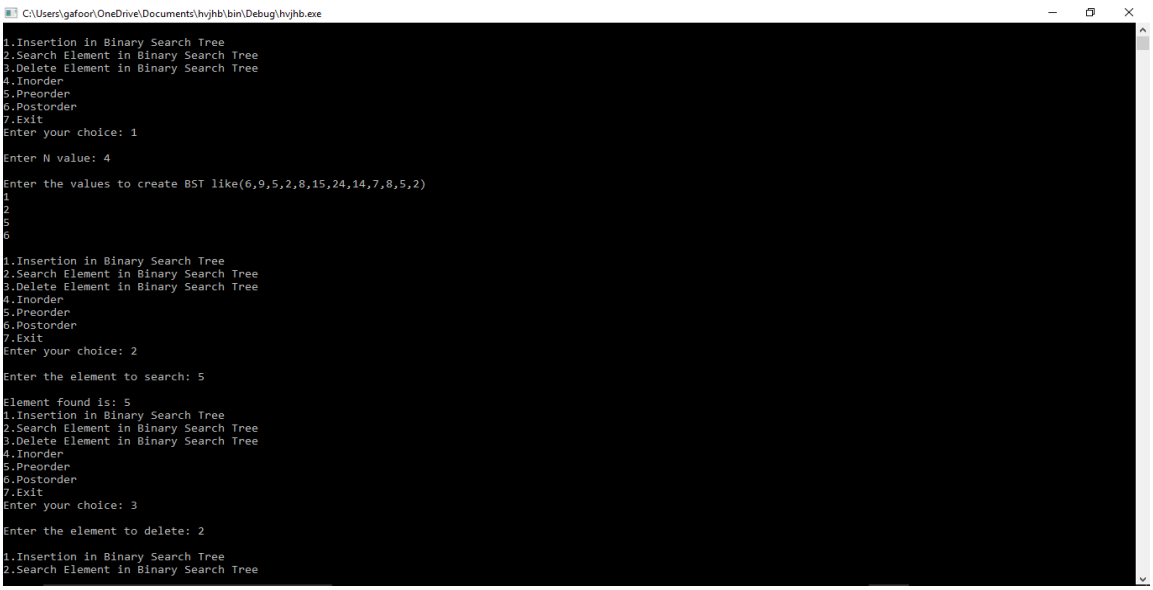
```
node = node->right;

else if(node->right == NULL)

node = node->left;

free(temp); /* temp is longer required */

}

}

return node;

}

void main()

{

int data, ch, i, n;

NODE *root=NULL;

while (1)

{

printf("\n1.Insertion in Binary Search Tree");

printf("\n2.Search Element in Binary Search Tree");
```

```
printf("\n3.Delete Element in Binary Search Tree");



printf("\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1: printf("\nEnter N value: " );

scanf("%d", &n);

printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\n");

for(i=0; i<n; i++)

{

scanf("%d", &data);

root=createtree(root, data);

}

break;
```

```
case 2: printf("\nEnter the element to search: ");

scanf("%d", &data);

root=search(root, data);

break;

case 3: printf("\nEnter the element to delete: ");

scanf("%d", &data);

root=del(root, data);

break;

case 4: printf("\nInorder Traversal: \n");

inorder(root);

break;

case 5: printf("\nPreorder Traversal: \n");

preorder(root);

break;

case 6: printf("\nPostorder Traversal: \n");

postorder(root);
```

```
break;

case 7: exit(0);

default:printf("\nWrong option");

break;

}

}

}
```

Output:




C:\Users\gafoor\OneDrive\Documents\hvjhb\bin\Debug\hvjhb.exe
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Delete Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 2
Enter the element to search: 5
Element found is: 5
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Delete Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 3
Enter the element to delete: 2
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Delete Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: 5
Preorder Traversal:
1 5 6
1.Insertion in Binary Search Tree
2.Search Element in Binary Search Tree
3.Delete Element in Binary Search Tree
4.Inorder
5.Preorder
6.Postorder
7.Exit
Enter your choice: