4.5 Binary Search Trees- Insertion, Deletion, Search

Assignment

```
#include <stdlib.h>
#include<stdio.h>
Struct tree
{
Int info;
Struct tree *left;
Struct tree *right;
 };
Struct tree *insert(struct tree *,int);
Void inorder(struct tree *);
Void postorder(struct tree *);
Void preorder(struct tree *);
Struct tree *delet(struct tree *,int);
Struct tree *search(struct tree *);
Int main(void)
{
Struct tree *root;
Int choice, item,item_no;
Root = NULL;
// rear = NULL;
Do {
Do {
Printf("\n \t 1. Insert in Binary Tree ");
Printf("\n\t 2. Delete from Binary Tree ");
```

```
Printf(“\n\t 3. Inorder traversal of Binary tree”);
Printf(“\n\t 4. Search”);
Printf(“\n\t 5. Exit “);
Printf(“\n\t Enter choice : “);
Scanf(“ %d”,&choice);
If(choice<1 || choice>7)
Printf(“\n Invalid choice – try again”);
 }
While (choice<1 || choice>7);
Switch(choice)
 {
 Case 1:
 Printf(“\n Enter new element: “);
 Scanf(“%d”, &item);
 Root= insert(root,item);
 Printf(“\n root is %d”,root->info);
 Printf(“\n Inorder traversal of binary tree is : “);
 Inorder(root);
 Break;
 Case 2:
 Printf(“\n Enter the element to be deleted : “);
 Scanf(“ %d”,&item_no);
 Root=delet(root,item_no);
 Inorder(root);
 Break;
 Case 3:
 Printf(“\n Inorder traversal of binary tree is : “);
 Inorder(root);
 Break;
```

```
 Case 4:
 Printf("\n Search operation in binary tree ");
 Root=search(root);
 Break;
 Default:
 Printf("\n End of program ");
 }
}
While(choice !=5);
Return(0);
}
Struct tree *insert(struct tree *root, int x)
 {
If(!root)
 {
 Root=(struct tree*)malloc(sizeof(struct tree));
 Root->info = x;
 Root->left = NULL;
 Root->right = NULL;
 Return(root);
 }
If(root->info > x)
Root->left = insert(root->left,x); else {
If(root->info < x)
Root->right = insert(root->right,x);
}
Return(root);
}
Void inorder(struct tree *root) {
```

```
If(root != NULL) {
Inorder(root->left);
Printf(“ %d”,root->info);
Inorder(root->right);
}
Return;
}
Struct tree *delet(struct tree *ptr,int x) {
Struct tree *p1,*p2;
If(!ptr) {
Printf(“\n Node not found “);
Return(ptr);
} else {
If(ptr->info < x) {
Ptr->right = delet(ptr->right,x);
/*return(ptr);*/
} else if (ptr->info >x) {
Ptr->left=delet(ptr->left,x);
Return ptr;
} else
{
If(ptr->info == x)
{
If(ptr->left == ptr->right)
{
Free(ptr);
Return(NULL);
} else if(ptr->left==NULL)
{
```

```
P1=ptr->right;
Free(ptr);
Return p1;
} else if(ptr->right==NULL)
{
P1=ptr->left;
Free(ptr);
Return p1;
} else {
P1=ptr->right;
P2=ptr->right;
While(p1->left != NULL)
P1=p1->left;
P1->left=ptr->left;
Free(ptr);
Return p2;
}
}
}
}
Return(ptr);
}
Struct tree *search(struct tree *root) {
Int no,I,ino;
Struct tree *ptr;
Ptr=root;
Printf("\n Enter the element to be searched :");
Scanf(" %d",&no);
Fflush(stdin);
```

```
While(ptr) {
If(no>ptr->info)
Ptr=ptr->right; else if(no<ptr->info)
Ptr=ptr->left; else
Break;
}
If(ptr) {
Printf("\n Element %d which was searched is found and is = %d",no,
Ptr->info);
} else
Printf("\n Element %d does not exist in the binary tree",no);
Return(root);
}
```

Output

```
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 1

Enter new element: 2

root is 2
Inorder traversal of binary tree is :  2
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 1

Enter new element: 4

root is 2
Inorder traversal of binary tree is :  2 4
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 1

Enter new element: 5

root is 2
Inorder traversal of binary tree is :  2 4 5
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 3

Inorder traversal of binary tree is :  2 4 5
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 4

Search operation in binary tree
Enter the element to be searched :4
```

```
Element 4 which was searched is found and is = 4
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 2

Enter the element to be deleted : 2
4 5
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice : 3

Inorder traversal of binary tree is :  4 5
 1. Insert in Binary Tree
 2. Delete from Binary Tree
 3. Inorder traversal of Binary tree
 4. Search
 5. Exit
 Enter choice :
```