NAME - AKRITI CHOUDHARY

ROLL NUMBER - 2005776

SUBJECT - DSA LAB

DATE - 2/11/2021

CLASS - B14

BRANCH - CSE

Question 1)WAP to create a binary search tree and traverse the tree with in-order, pre order and post order by proving suitable menu for the user.

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
   int data;
   struct Node *left;
   struct Node *right;
};
struct Node *newNode(int item)
{
   struct Node *temp = (struct Node *)malloc(sizeof(struct Node));
   temp->data = item;
   temp->left = temp->right = NULL;
   return temp;
}
struct Node *create(struct Node *root, int val)
{

   if (root == NULL)
   {
      return newNode(val);
   }

   if (val < root->data)
   {
      root->left = create(root->left, val);
   }
   else
   {
      root->right = create(root->right, val);
   }
   return root;
}
void inorder(struct Node *root)
{

   if (root == NULL)
   {
      return;
   }
   inorder(root->left);
   printf("%d ", root->data);
```

```
    inorder(root->right);
}
void preorder(struct Node *root)
{

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

    if (root == NULL)
    {
        return;
    }
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
int main()
{
    struct Node *root = NULL;
    int n;
    int a;
    int b;
    printf("BST-");
    do{
    printf("\n1 -> Create\n2 -> In-order\n3 -> Postorder\n4 -> Preorder\n5 -> Quit");
    printf("\nEnter your choice : ");
    scanf("%d", &b);
    switch (b)
    {
    case 1:

        printf("\nEnter total numbers of nodes to be created : ");
        scanf("%d", &n);
        printf("\nEnter root : ");
        scanf("%d", &a);
        root = create(root, a);
        for (int i = 0; i < n - 1; i++)
        {
            printf("\nEnter value of root to be inserted : ");
```

```
        scanf("%d", &a);
        create(root, a);
      }
      break;
    case 2:
      inorder(root);
      break;
    case 3:
      postorder(root);
      break;
    case 4:
      preorder(root);
      break;
    case 5:
      puts("--------Program Terminated-------");
      break;
    default:
      break;
    }
    }while(b!=5);
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\2_11_2021> ./tree_order
BST-
1 -> Create
2 -> In-order
3 -> Postorder
4 -> Preorder
5 -> Quit
Enter your choice : 1

Enter total numbers of nodes to be created : 5

Enter root : 1

Enter value of root to be inserted : 2

Enter value of root to be inserted : 3

Enter value of root to be inserted : 4

Enter value of root to be inserted : 5

1 -> Create
```

```
1 -> Create
2 -> In-order
3 -> Postorder
4 -> Preorder
5 -> Quit
Enter your choice : 2
1 2 3 4 5
1 -> Create
2 -> In-order
3 -> Postorder
4 -> Preorder
5 -> Quit
Enter your choice : 3
5 4 3 2 1
1 -> Create
2 -> In-order
3 -> Postorder
4 -> Preorder
5 -> Quit
Enter your choice : 4
1 2 3 4 5
1 -> Create
2 -> In-order
3 -> Postorder
4 -> Preorder
5 -> Quit
Enter your choice : 5
--------Program Terminated-------
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\2_11_2021>
```

Question 2) Modify the program LE8.1 by providing suitable user define functions separately to find the smallest and largest elements in the binary search tree.

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

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

struct Node *createNode(int data)
{
	struct Node *n;
	n = (struct Node *)malloc(sizeof(struct Node));

	n->data = data;
	n->left = NULL;
	n->right = NULL;
}

struct Node *insert(struct Node *root, int val)
{
	if (root == NULL)
	{
		return createNode(val);
	}

	if(val == root->data)
	{
		puts("BST Tree cannot have duplicate values");
	}

	if (val < root->data)
	{
		root->left = insert(root->left, val);
	}

	if (val > root->data)
	{
		root->right = insert(root->right, val);
	}
```

```
	return root;
}

void smallest(struct Node *root)
{
	while (root->left != NULL)
	{
		root = root->left;
	}

	printf("Smallest element of the Tree = %d \n", root->data);
}

void largest(struct Node *root)
{
	while (root->right != NULL)
	{
		root = root->right;
	}

	printf("Largest element of the Tree = %d \n", root->data);
}

void preOrder(struct Node *root)
{

	if (root != NULL)
	{
		printf("%d ", root->data);
		preOrder(root->left);
		preOrder(root->right);
	}
}

void inOrder(struct Node *root)
{

	if (root != NULL)
	{
		inOrder(root->left);
		printf("%d ", root->data);
		inOrder(root->right);
	}
}

void postOrder(struct Node *root)
{
```

```
    if (root != NULL)
    {
        postOrder(root->left);
        postOrder(root->right);
        printf("%d ", root->data);
    }
}

int main()
{

    int choice, i, n, num, c;

    struct Node *root = NULL;

    do
    {
        printf("Binary Search Tree ( BST ) Menu \n");
        printf("0. Quit \n");
        printf("1. Create \n");
        printf("2. Inorder Traversal \n");
        printf("3. PreOrder Traversal \n");
        printf("4. PostOrder Traversal \n");
        printf("5.Find smallest element \n");
        printf("6. Find largest element \n");

        printf("Enter your choice : \n");
        scanf("%d", &choice);

        switch (choice)
        {

        case 0:
            exit(0);
            break;

        case 1:
            printf("Enter the number of Nodes : \n");
            scanf("%d", &n);

            printf("Enter %d elements : \n", n);

            for (i = 1; i <= n; i++)
            {
                scanf("%d", &num);

                root = insert(root, num);
```

```
        }

        break;

case 2:
        if (root == NULL)
                printf("Tree is Empty \n");

        else
        {
                printf("Inorder Traversal : \n");
                inOrder(root);
        }
        printf("\n");
        break;

case 3:
        if (root == NULL)
                printf("Tree is Empty \n");

        else
        {
                printf("PreOrder Traversal : \n");
                preOrder(root);
        }
        printf("\n");
        break;

case 4:
        if (root == NULL)
                printf("Tree is Empty \n");

        else
        {
                printf("PostOrder Traversal : \n");
                postOrder(root);
        }
        printf("\n");
        break;

case 5:
        smallest(root);
        break;

case 6:
        largest(root);
        break;
```

```
            default:
                printf("Wrong input .... \n");
            }

        } while (1);

        return 0;
}
```

```
Binary Search Tree ( BST ) Menu
0. Quit
1. Create
2. Inorder Traversal
3. PreOrder Traversal
4. PostOrder Traversal
5.Find smallest element
6. Find largest element
Enter your choice :
1
Enter the number of Nodes :
5
Enter 5 elements :
1
2
3
4
5
Binary Search Tree ( BST ) Menu
0. Quit
1. Create
2. Inorder Traversal
3. PreOrder Traversal
4. PostOrder Traversal
5.Find smallest element
6. Find largest element
5.Find smallest element
6. Find largest element
Enter your choice :
5
Smallest element of the Tree = 1
Binary Search Tree ( BST ) Menu
0. Quit
1. Create
2. Inorder Traversal
3. PreOrder Traversal
4. PostOrder Traversal
5.Find smallest element
6. Find largest element
Enter your choice :
6
Largest element of the Tree = 5
Binary Search Tree ( BST ) Menu
0. Quit
1. Create
2. Inorder Traversal
3. PreOrder Traversal
4. PostOrder Traversal
5.Find smallest element
6. Find largest element
Enter your choice :
0
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\2_11_2021>
```

Question 3) Modify the program LE8.2 by providing suitable user defined functions to insert and delete elements from the binary search tree.

```
#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;
}
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 *smallest(NODE *node)
{
    while (node != NULL && node->left != NULL)
        node = node->left;
    printf("smallest value is %d\n", node->data);
    return node;
}
void largest(NODE *node)
{
    while (node != NULL && node->right != NULL)
        node = node->right;
    printf("largest value is %d\n", node->data);
}
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
    {
        if (node->right && node->left)
```

```
    {
      temp = smallest(node->right);
      node->data = temp->data;
      node->right = del(node->right, temp->data);
    }
    else
    {
      temp = node;
      if (node->left == NULL)
        node = node->right;
      else if (node->right == NULL)
        node = node->left;
      free(temp);
    }
  }
  return node;
}
NODE *insert(NODE *node, int key)
{
  if (node == NULL)
  {
    return createtree(node, key);
  }
  if (key < node->data)
  {
    node->left = insert(node->left, key);
  }
  else if (key > node->data)
  {
    node->right = insert(node->right,key);
  }
  return node;
}
  int main()
  {
    int data, ch, i, n;
    NODE *root = NULL;
    do
    {
      printf("\n----------------------");
      printf("Binary Search Tree Menu");
      printf("----------------------\n");
      printf("\n0. Quit");
      printf("\n1. Create");
      printf("\n2. Inorder Traversal");
      printf("\n3. Preorder Traversal");
      printf("\n4. Postorder Traversal");
```

```
printf("\n5. Find smallest element");
printf("\n6. Find largest element");
printf("\n7. Insert an element");
printf("\n8. Delete an element");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch (ch)
{
case 0:
   printf("-------Program Terminated---------\n");
   break;
case 1:
   printf("\nEnter number of nodes: ");
   scanf("%d", &n);
   printf("\nEnter %d elements \n", n);
   for (i = 0; i < n; i++)
   {
      scanf("%d", &data);
      root = createtree(root, data);
   }
   printf("The BST is created\n");
   break;
case 2:
   printf("\nThe Inorder Traversal is: ");
   inorder(root);
   break;
case 3:
   printf("\nThe Preorder Traversal is: ");
   preorder(root);
   break;
case 4:
   printf("\nThe Postorder Traversal is: ");
   postorder(root);
   break;
case 5:
   smallest(root);
   break;
case 6:
   largest(root);
   break;
case 7:  printf("\nEnter the element to insert: ");
    scanf("%d", &data);
    root=insert(root, data);
   break;
case 8:
   printf("\nEnter the element to delete: ");
   scanf("%d", &data);
```

```
            root = del(root, data);
            break;
        default:
            printf("\nWrong option");
            break;
        }
    } while (ch != 0);
}
```

```
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\2_11_2021> ./DSA_Q3

----------------------Binary Search Tree Menu----------------------

0. Quit
1. Create
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Find smallest element
6. Find largest element
7. Insert an element
8. Delete an element
Enter your choice: 1

Enter number of nodes: 3

Enter 3 elements
4 2 1
The BST is created

----------------------Binary Search Tree Menu----------------------

0. Quit
1. Create
2. Inorder Traversal
```

```
3. Preorder Traversal
4. Postorder Traversal
5. Find smallest element
6. Find largest element
7. Insert an element
8. Delete an element
Enter your choice: 8

Enter the element to delete: 2

----------------------Binary Search Tree Menu----------------------

0. Quit
1. Create
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Find smallest element
6. Find largest element
7. Insert an element
8. Delete an element
Enter your choice: 2

The Inorder Traversal is: 1     4
----------------------Binary Search Tree Menu----------------------

0. Quit
1. Create
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Find smallest element
6. Find largest element
7. Insert an element
8. Delete an element
Enter your choice: 0
-------Program Terminated---------
PS D:\KIIT_NOTES\2nd year sem_3\dsa_lab\2_11_2021>
```