Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .

- Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
- Traverse the BST in Inorder, Preorder and Post Order
- Search the BST for a given element (KEY) and report the appropriate message
- Exit

Student Name: Neha Sharma **UID:20BCS4576**
Branch: IOT **Section/Group A**
Semester: 3rd **Date of Performance:12/11/2021**
Subject Name: Data Structure Lab **Subject Code:** 20CSP-236_20BIT-1_A

1. Aim/Overview of the practical: Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .

2. Task to be done:

1. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
2. Traverse the BST in Inorder, Preorder and Post Order
3. Search the BST for a given element (KEY) and report the appropriate message
4. Exit

3. Algorithm/Flowchart:

Step 1: Start.

Step 2: Create a Binary Search Tree for N elements.

Step 3: Traverse the tree in inorder.

Step 4: Traverse the tree in pre order

Step 6: Traverse the tree in post order.

Step 7: Search the given key element in the BST.

Step 8: Delete an element from BST.

Step 9: Stop

5. **Steps for experiment/practical(code):**

```
#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 if(node->right && node->left){

temp = findMin(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); /* temp is longer required */

}

return node;
```

```
}

int 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;

 exit(0);
```

```
default:printf("\nWrong option");

break;

}

}

return 0 ;

}
```

5. Output: Image of sample output to be attached here

```
C:\Users\hp\Desktop\DS LAB WS\tree program.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: 1

Enter N value: 11

Enter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)
50 17 12 9 14 23 19 72 54 67 76

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: 4

Inorder Traversal:
9       12      14      17      19      23      50      54      67      72      76
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:
50      17      12      9       14      23      19      72      54      67      76
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: 6

Postorder Traversal:
9       14      12      19      23      17      67      54      76      72      50
```

Learning outcomes (What I have learnt):

1.

2.

3.

4.

5.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			