PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.04

- **Aim:**

Implementation of operations on Binary Search Tree.

- **Prerequisite:**

C programming

- **Outcome:**

After successful completion of this experiment students will be,

Apply concepts of Trees and Graphs to a given problem.

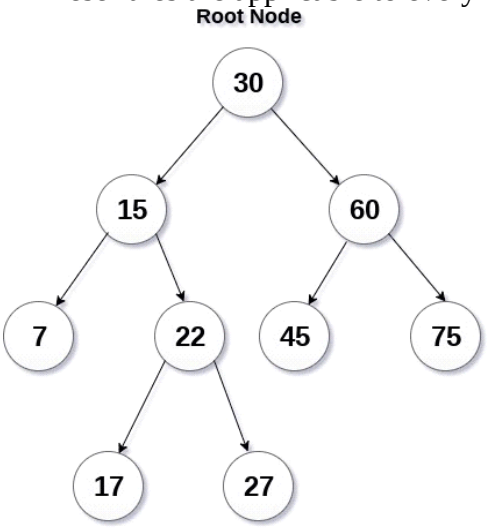
- **Theory:**

Binary Search Tree:

A binary search tree is an ordered binary tree. A Binary Search Tree (BST) is a tree in which all the nodes follow following properties –

All the nodes in the left sub-tree have a value less than that of the root (parent) node. All the nodes in the right sub-tree have a value either equal to or greater than that of the root (parent) node.

These rules are applicable to every subtree in a tree.



Binary Search Tree

Node Structure:

```
struct BSTnode
{
        int data;
        struct BSTnode *left;
        struct BSTnode *right;
}
```

Algorithm:

- **Create BST and Insert a node:**

Step 1: Call malloc to allocate memory to newnode
Step 2: Set newnode->data and newnode->left = newnode->right = NULL
Step 3: if root = NULL then
root = temp = newnode
return root
else
repeat step 4 while temp != NULL
Step 4: if newnode->data <= temp->data if
temp->left == NULL
temp->left = newnode return
root
temp = temp->left

Step 5: Exit

else

if temp->right == NULL
temp->right = newnode return root
temp = temp->right

. **Display: (Inorder Traversal)**

Step 1: IF root != NULL then
Step 2: Inorder(root->left) Step
3: Print root->data
Step 4: Inorder(root->right) [End
of Loop]
Step 5: Stop

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

Roll. No. A56	Name: ANAS MUBEEN EDROOS
Class: DSE-MTRX	Batch: DSE
Date of Experiment: 6/3/2021	Date of Submission: 13/3/2021
Grade:	

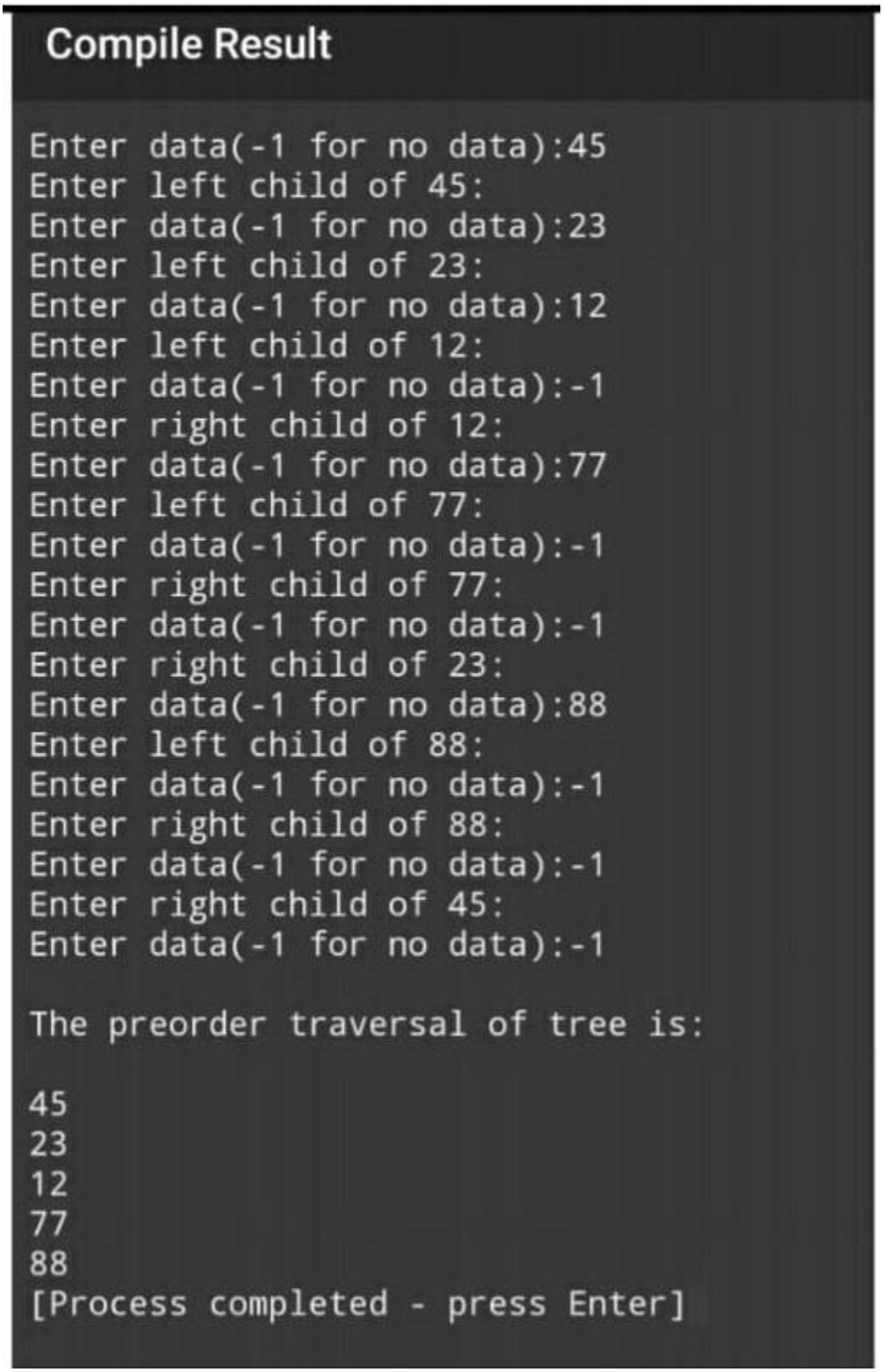
B1. Software Code written by student:

Program:-

```
//Anas Edroos
//Roll No 56
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
int data;
```

```
struct node *left;
struct node *right;
} node;
node *create()
{
node *p;
int x;
printf("Enter data(-1 for no data):");
scanf("%d",&x);
if(x==-1)
return NULL;
p=(node*)malloc(sizeof(node));
p->data=x;
printf("Enter left child of %d:\n",x);
p->left=create();
printf("Enter right child of %d:\n",x);
p->right=create();
return p;
}
void preorder(node *t)
{
if(t!=NULL)
{
printf("\n%d",t->data);
preorder(t->left);
preorder(t->right);
}
}
int main()
{
node *root;
root=create();
printf("\nThe preorder traversal of tree is:\n");
preorder(root);
return 0;
}
```

B2. Output:



B3. Observations and learning:

We observe how operations are performed on Binary search tree.

B4. Conclusion:

We have understood the implementation of Binary Tree.

B5. Question of Curiosity

1) What is binary tree and Binary search tree?

soln. IN BINARY TREE there is no ordering in terms of how the nodes are arranged

IN BINARY SEARCH TREE the left subtree has elements less than the nodes element and

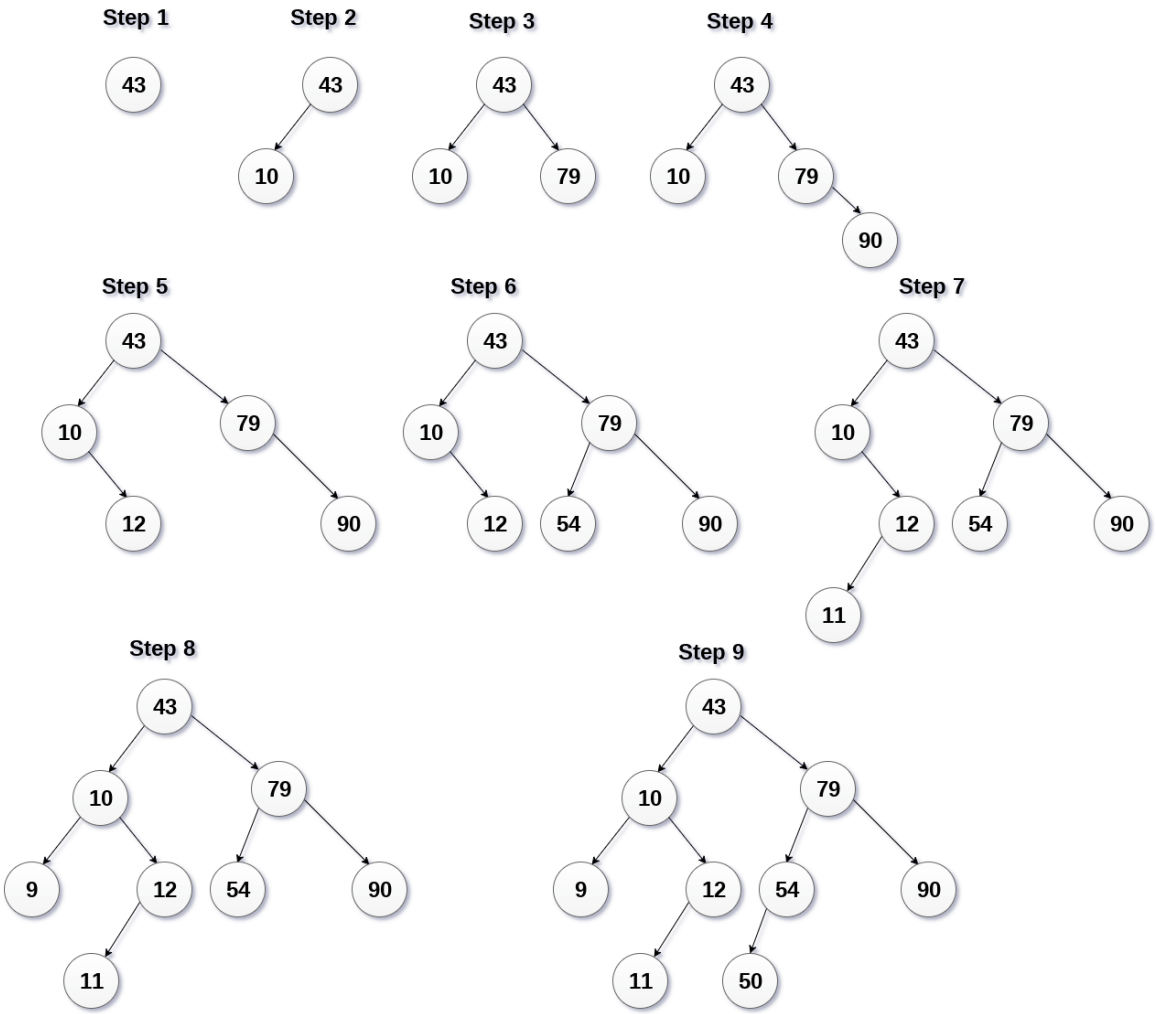
the right subtree has elements greater than the nodes element

2) Write an algorithm to search an element in a binary search tree

soln. Step 1: low = 0
Step2: high = n-1
Step 3: while (low <= high) repeat
step 4 through step 6 Step 4: mid = (low + high) / 2
Step 5: Is (ele = = a[mid]) then Loc = mid goto step 7 Otherwise
Step 6: is (ele < a[mid])? then high = mid – 1 otherwise low = mid + 1 [end while – step 3]
Step 7: Is (Loc >= 0) ? then Print “search element found at location “, loc Otherwise Print “search element not found”.

3) Construct a Binary search tree by inserting 43, 10, 79, 90, 12, 54, 11, 9, 50

Soln.



Binary search Tree Creation