Problem Statement

Write a Program to implement following operations in Binary Search Tree.

- ➢ Create a Binary Search Tree
- ➢ Traverse a Tree in in-order
- ➢ Traverse a tree in post-order
- ➢ Traverse a tree in pre-order

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct Tree {
	int data;
	Tree *left;
	Tree *right;
};
typedef struct Tree tree;
tree *root;
void insert(int x);
tree* create_node(int x);
void remove();
void traverse_preorder(tree *t);
void traverse_postorder(tree *t);
void traverse_inorder(tree *t);
int main() {
	root = create_node(10);
	insert(20);
	insert(8);
	insert(7);
	insert(15);
	insert(17);
	insert(4);
	insert(1);
	cout<<endl<<"Preorder Traversal: "<<endl;
	traverse_preorder(root);
	cout<<endl<<"Postorder Traversal: "<<endl;
	traverse_postorder(root);
	cout<<endl<<"Inorder Traversal: "<<endl;
	traverse_inorder(root);
	return 0;
}
tree* create_node(int x) {
	tree *newnode = (tree*)malloc(sizeof(tree));
	newnode->data = x;
	newnode->left = NULL;
	newnode->right = NULL;
```

```
	return newnode;
}
void insert(int x) {
	tree *t,*t1;
	tree *newnode = create_node(x);
	t = root;
	while(t) {
		if(x < t->data) {
			t1 = t;
			t = t->left;
		}

		else {
			t1 = t;
			t = t->right;
		}

	}
	if(x < t1->data)
	t1->left = newnode;
	else
	t1->right = newnode;
}
void traverse_preorder(tree *t) {
	if(t) {
		cout<<t->data<<", ";
		traverse_preorder(t->left);
		traverse_preorder(t->right);
	}

}
void traverse_postorder(tree *t) {
	if(t) {

		traverse_postorder(t->left);
		traverse_postorder(t->right);
		cout<<t->data<<", ";
	}

}
void traverse_inorder(tree *t) {
	if(t) {

		traverse_inorder(t->left);
		cout<<t->data<<", ";
		traverse_inorder(t->right);
	}
	}
```