1)
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Binarytree
{
    int data;
    struct Binarytree *l;
    struct Binarytree *r;
} nod;

nod *create();
void insert(nod *, nod *);
void preorder(nod *);
void postorder(nod *);
void inorder(nod *);

int main()
{
    int var;
    nod*root = NULL, *temp, *current;
    printf("Enter the number of Nodes you want to be in binarytree :");
    scanf("%d", &var);
    printf("Enter %d Nodes data ",var);
    do
    {
        temp = create();
        if (root == NULL)
            root = temp;
        else
            insert(root, temp);
        var--;

    } while (var != 0);

    printf("Preorder");
    preorder(root);

    printf("Inorder");
    inorder(root);

    printf("Postorder");
    postorder(root);
    return 0;
}

nod *create()
{
    nod *ano;

    ano= (nod *)malloc(sizeof(nod));
    scanf("%d", &ano->data);
    ano->l = ano->r = NULL;
    return ano;
}

void insert(nod *root, nod *tre)
{
    if (root == NULL)
    {
        root = tre;
    }
    else
    {

        if (tre->data < root->data)
        {
            if (root->l != NULL)
                insert(root->l, tre);
            else
                root->l = tre;
        }

        if (tre->data > root->data)
        {
            if (root->r != NULL)
                insert(root->r, tre);
            else
                root->r = tre;
        }
    }
}

void preorder(nod *root)
```

```
{
    if (root != NULL)
    {
        printf("%d  ", root->data);
        preorder(root->l);
        preorder(root->r);
    }
}
```

2)

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

struct node
{
    int data; //node will store an integer
    struct node *right_child; // right child
    struct node *left_child; // left child
};
struct node* insert(struct node *root, int x)
{
    if(root==NULL)
        return x;
    else if(x>root->data)
        root->right_child = insert(root->right_child, x);
    else
        root->left_child = insert(root->left_child,x);
    return root;
}

// funnction to delete a node


void inorder(struct node *root)
{
    if(root!=NULL) // checking if the root is not null
    {
        inorder(root->left_child); // visiting left child
        printf(" %d ", root->data); // printing data at root
        inorder(root->right_child);// visiting right child
    }
}

int main()
{
    struct node *root;
    root=20;
    insert(root,5);
    insert(root,1);
    insert(root,15);
    insert(root,9);
    insert(root,7);
    insert(root,12);
    insert(root,30);
    insert(root,25);
    insert(root,40);
    insert(root, 45);
    insert(root, 42);

    inorder(root);
    printf("\n");

}
```

3)

```
#include <stdio.h>

int main()
{

    int number,x,k, val_find, found = 0;
    printf("Enter the number of elements that u want to be in the array: ");
    scanf("%d", &number);
    int arr[number];
    printf("Enter the elements sequentially: \n");
    for (k = 0; k< number; k++)
    {
        scanf("%d", &arr[k]);
    }

    printf("Enter the element to be searched: ");
    scanf("%d", &val_find);

    for (x = 0; x< number; x++)
    {
        if (val_find == arr[x])
        {
            found = 1;
            break;
        }
    }
    if (found == 1)
        printf("Element is there in the array at the position %d", x );
    else
        printf("Element isn't  there in the array\n");

    return 0;
}
```

4)

```
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
        return binarySearch(arr, mid + 1, r, x);
    }
    return -1;
}

void main(void)
{
    int arr[] = { 2,1,5,45,23,67,};
    int n = sizeof(arr) / sizeof(arr[0]);
    int x = 10;
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? printf("Element is not present in array")
```

```
                        : printf("Element is present at index %d", result);
}
```