

## **1. Write a C program to print preorder, inorder, and postorder traversal on Binary Tree.**

### **Code:**

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* left,* right;
};
struct node* nNode(int data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return(node);
}
void printPostorder(struct node* node)
{
    if (node == NULL)
        return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d ", node->data);
}
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}
```

```

int main()
{
    struct node *root = nNode(9);
    root->left = nNode(8);
    root->right = nNode(7);
    root->left->left = nNode(6);
    root->left->right = nNode(1);
    printf("\nPreorder traversal of binary tree is \n");
    printPreorder(root);
    printf("\nInorder traversal of binary tree is \n");
    printInorder(root);
    printf("\nPostorder traversal of binary tree is \n");
    printPostorder(root);
    return 0;
}

```

---

## **2) Write a C program to create (or insert) and inorder traversal on Binary Search Tree**

### **Code:**

```

#include<stdio.h>
#include<stdlib.h>
struct btnode
{
    int value;
    struct btnode *left;
    struct btnode *right;
} *root = NULL, *temp = NULL, *x2, *x1;
int insert();
int inorder(struct btnode *x);
int flag = 1;
int main()
{
    int ch;
    printf("\n1) Insert an element into the tree: \n2) Inorder Traversal: \n3) Exit: \n");
    while(1)
    {
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                insert();
                break;
            case 2:
                inorder(root);
                break;
            case 3:
                return 0;
            default:
                continue;
        }
    }
}

```

```

break;
case 3:
exit(0);
default :
printf("Invalid Entry, Enter the choice again: ");
break;
}
}
return 0;
}
int insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
    return 0;
}
int create()
{
    int data;
    printf("Enter the data node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->rule = temp->r = NULL;
    return 0;
}
int search(struct btnode *x)
{
    if ((temp->value > x->value) && (x->r != NULL))
        search(x->r);
    else if ((temp->value > x->value) && (x->r == NULL))
        x->r = temp;
    else if ((temp->value < x->value) && (x->rule != NULL))
        search(x->rule);
    else if ((temp->value < x->value) && (x->rule == NULL))
        x->rule = temp;
    return 0;
}
int inorder(struct btnode *x)
{
    if (root == NULL)
    {
        printf("Sorry! There are No elements to display");
    }
}

```

```

return;
}
if (x->rule != NULL)
inorder(x->rule);
printf("%d -> ", x->value);
if (x->r != NULL)
inorder(x->r);
return 0;
}

```

### **3. Write a C program for the linear search algorithm.**

#### **Code:**

```

#include <stdio.h>

void main()
{ int num;

    int i, num_s, flag = 0;

    printf("Enter the number of elements ");
    scanf("%d", &num);
    int array[num];
    printf("Enter the elements one by one \n");
    for (i = 0; i < num; i++)
    {
        scanf("%d", &array[i]);
    }

    printf("Enter the element to be searched ");
    scanf("%d", &num_s);
    /* Linear search begins */
    for (i = 0; i < num ; i++)
    {
        if (num_s == array[i] )
        {
            flag = 1;
            break;
        }
    }
    if (flag == 1)
        printf("Element is present in the array at position
%d", i+1); else
        printf("Element is not present in the array\n");
}

```

#### **4)Write a C program for binary search algorithm**

##### **Code:**

```
#include <stdio.h>
int main()
{
    int i, initial, max, middle, x, search, array[50];

    printf("Enter no.of elements you want to enter in the array: ");
    scanf("%d", &x);

    printf("Enter %d integers: ", x);

    for (i = 0; i < x; i++)
        scanf("%d", &array[i]);

    printf("Enter the value for to search: ");
    scanf("%d", &search);

    initial = 0;
    max = x - 1;
    middle = (initial+max)/2;

    while (initial <= max) {
        if (array[middle] < search)
            initial = middle + 1;
        else if (array[middle] == search) {
            printf("%d found at the location %d.\n", search,
                middle+1); break;
        }
        else
            max = middle - 1;

        middle = (initial + max)/2;
    }
    if (initial > max)
        printf("Invalid Entry %d not present in the list.\n", search);

    return 0;
}
```