# Lab Record

**Question 1: Write a C program to print pre-order, in-order, and post-order traversal on Binary Tree.**

Answer:

**Code:**

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 int data;
 struct node* left;
 struct node* right;
};
void inorder(struct node* root)
{
 if(root == NULL)
return;
 inorder(root->left);
 printf("%d ", root->data);
 inorder(root->right);
}
void preorder(struct node* root)
{
 if(root == NULL)
return;
```

```
printf("%d ", root->data);
preorder(root->left);
preorder(root->right);
}
void postorder(struct node* root)
{
 if(root == NULL)
 return;
 postorder(root->left);
 postorder(root->right);
 printf("%d ", root->data);
}
struct node* createNode(value)
{
 struct node* newNode = malloc(sizeof(struct node));
 newNode->data = value;
 newNode->left = NULL;
 newNode->right = NULL;
return newNode;
}
struct node* insertL(struct node *root, int value)
 {
 root->left = createNode(value);
 return root->left;
}

struct node* insertR(struct node *root, int value)
```

```
{
 root->right = createNode(value);
 return root->right;
}
void main()
{
 struct node* root = createNode(1000);
 insertL(root, 500);
 insertR(root, 2000);
 insertL(root->left, 200);
 insertR(root->left,800 );
 printf("Inorder traversal is following\n");
 inorder(root);
 printf("\nPreorder traversal is following\n");
 preorder(root);

 printf("\nPostorder traversal is following \n");
 postorder(root);
}
```

**Output:**

Inorder traversal is following

200 500 800 1000 2000

Preorder traversal is following

1000 500 200 800 2000

Postorder traversal is following

200 800 500 2000 1000

**Questions 2: Write a C program to create (or insert) and in-order traversal on Binary Search Tree.**

Answer:

**Code:**

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

struct node
{
int data;
struct node *left, *right;
};

struct node *newNode(int item)
{
struct node *temp = (struct node *)malloc(sizeof(struct node));
temp->data = item;
temp->left = temp->right = NULL;
return temp;
}

void inorder(struct node *root)
{
if (root != NULL)
{
inorder(root->left);
```

```
printf("%d \t", root->data);
inorder(root->right);
}
}

struct node* insert(struct node* node, int data)
{
if (node == NULL)
return newNode(data);
if (data< node->data)
node->left  = insert(node->left, data);
else if (data > node->data)
node->right = insert(node->right, data);
return node;
}

void main()
{
struct node *root = NULL;
root = insert(root, 1000);
insert(root, 500);
insert(root, 200);
insert(root, 800);
insert(root, 2000);
insert(root, 1500);
insert(root, 4000);
printf("Inorder Traversal is following \n");
```

```
inorder(root);
}
```

**Output:**

Inorder traversal is following

200 500 800 1000 1500 2000 4000

**Question 3: Write a C program for linear search algorithm.**

Answer:

**Code:**

```
#include<stdio.h>
#include<stdlib.h>
int search(int a[5], int p)
{
int i;
for (i=0; i<5; i++)
{
if (a[i]==p)
return i;
}
return -1;
}

void main()
{
int p,i,a[5];
printf("Enter five numbers in the array \n");
```

```
for(i=0;i<5;i++)
{
scanf("%d\n",&a[i]);
}
printf("Enter the number to be searched \t");
scanf("%d",&p);
int result = search(a,p);
if(result==-1)
printf("Element is not present in array");
else
printf("Element is present at position %d", result+1);
}
```

**Output:**

```
Enter five numbers in the array
1
2
3
4
5
Enter the number to be searched  4
Element is present at position  4
```

**Question 4: Write a C program for binary search algorithm**

Answer:

**Code:**

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

```
int binarySearch(int a[], int m, int r, int x)
{
if (r>= m)
{
int mid = m + (r-m)/2;
if (a[mid]==x)
return mid;
else if (a[mid]>x)
return binarySearch(a, m, mid - 1, x);
return binarySearch(a, mid + 1, r, x);
}
return -1;
}

void main()
{
int p,i,a[5];
printf("Enter five numbers in the array \n");
for(i=0;i<5;i++)
{
scanf("%d\n",&a[i]);
}
printf("Enter the number to be searched \t");
scanf("%d",&p);
int result = binarySearch(a, 0, 4, p);
if(result==-1)
printf("Element is not present in array");
```

```
else
printf("Element is present at position %d", result+1);
}
```

**Output:**

```
Enter five numbers in the array
1
2
3
4
5
Enter the number to be searched  4
Element is present at position  4
```

**********