## DSA LAB ASSIGNMENT

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

```
#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* insertLeft(struct node *root, int value) {
   root->left = createNode(value);
   return root->left;
}


struct node* insertRight(struct node *root, int value){
   root->right = createNode(value);
   return root->right;
}


int main(){
   struct node* root = createNode(1);
   insertLeft(root, 12);
   insertRight(root, 9);

   insertLeft(root->left, 5);
   insertRight(root->left, 6);

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

   printf("\nPreorder traversal \n");
   preorder(root);

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

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

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

struct btnode
{
   int value;
   struct btnode *l;
   struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;
```

```
void delete1();
void insert();
void delete();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void preorder(struct btnode *t);
void postorder(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);
int largest(struct btnode *t);

int flag = 1;

void main()
{
   int ch;

   printf("\nOPERATIONS ---");
   printf("\n1 - Insert an element iInto tree\n");
   printf("2 - Delete an element from the tree\n");
   printf("3 - Inorder Traversal\n");
   printf("4 - Preorder Traversal\n");
   printf("5 - Postorder Traversal\n");
   printf("6 - Exit\n");
   while(1)
   {
      printf("\nEnter your choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         insert();
         break;
      case 2:
         delete();
         break;
      case 3:
         inorder(root);
         break;
      case 4:
         preorder(root);
         break;
```

```
        case 5:
            postorder(root);
            break;
        case 6:
            exit(0);
        default :
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* To insert a node in the tree */
void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}

/* To create a node */
void create()
{
    int data;

    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}
```

**3.C program for linear search algorithm.**

```
#include <stdio.h>
int main()
{
```

```
  int array[100], search, c, n;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integer(s)\n", n);

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

  printf("Enter a number to search\n");
  scanf("%d", &search);

  for (c = 0; c < n; c++)
  {
    if (array[c] == search)    /* If required element is found */
    {
      printf("%d is present at location %d.\n", search, c+1);
      break;
    }
  }
  if (c == n)
    printf("%d isn't present in the array.\n", search);

  return 0;
}
```

**4.C program for binary search algorithm**

```
include <stdio.h>
int main()
{
  int c, first, last, middle, n, search, array[100];

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
```

```
  printf("Enter value to find\n");
  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;

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

    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d isn't present in the list.\n", search);

  return 0;
}
```