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;
  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(int value)
{
  struct node* newNode = malloc(sizeof(struct node));
  newNode->data = value;
  newNode->left = NULL;
  newNode->right = NULL;

    return newNode;

}

void main()
{
  struct node* root = createNode(1);
  root->left=createNode(12);
  root->right=createNode(9);
  root->left->left=createNode(10);
  root->left->right=createNode(15);

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

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

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

}
```

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

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

typedef struct BST
{
   int data;
   struct BST *lchild, *rchild;
} node;

void insert(node *, node *);
void inorder(node *);
void main()
{
   int choice;
   char ans = 'N';
   int key;
   node *new_node, *root, *tmp, *parent;
   node *get_node();
   root = NULL;


   printf("\nProgram For Binary Search Tree ");
   do {
     printf("\n1.Create");
     printf("\n2.In order traversal");
     printf("\n3.Exit");
     printf("\nEnter your choice :");
     scanf("%d", &choice);

     switch (choice) {
     case 1:
       do {
         new_node = get_node();
         printf("\nEnter The Element ");
         scanf("%d", &new_node->data);

         if (root == NULL) /* Tree is not Created */
           root = new_node;
         else
           insert(root, new_node);

         printf("\nWant To enter More Elements?(y/n)");
         ans = getch();
       } while (ans == 'y');
       break;
    case 2:
       if (root == NULL)
         printf("Tree Is Not Created");
       else {
         printf("\nThe Inorder display : ");
         inorder(root);

       }
```

```
            break;
        }
    } while (choice != 3);
}
/*
 Get new Node
 */
node *get_node() {
    node *temp;
    temp = (node *) malloc(sizeof(node));
    temp->lchild = NULL;
    temp->rchild = NULL;
    return temp;
}
/*
 This function is for creating a binary search tree
 */
void insert(node *root, node *new_node) {
    if (new_node->data < root->data) {
        if (root->lchild == NULL)
            root->lchild = new_node;
        else
            insert(root->lchild, new_node);
    }

    if (new_node->data > root->data) {
        if (root->rchild == NULL)
            root->rchild = new_node;
        else
            insert(root->rchild, new_node);
    }
}
/*
 This function displays the tree in inorder fashion
 */
void inorder(node *temp) {
    if (temp != NULL) {
        inorder(temp->lchild);
        printf("%d", temp->data);
        inorder(temp->rchild);
    }
}
```

3)Write a C program for binary search algorithm

CODE:

```
#include <stdio.h>
int main()
{
  int array[100], data, i, n;

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

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

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

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

  for (i = 0; i < n; i++)
  {
    if (array[i] == data)
    {
      printf("%d is present at location %d.\n", data, i+1);
      break;
    }
  }
  if (i == n)
    printf("%d isn't present in the array.\n", data);

  return 0;
}
```

4)Write a C program for linear search algorithm

CODE:

```
#include <stdio.h>
int main()
{
  int i, left, right, middle, n, data, array[100];

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

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

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

  left = 0;
   right = n - 1;
  middle = (left+ right)/2;

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

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

  return 0;
}
```