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

code:

```
#include <stdio.h>
#include <stdlib.h>
void Postorder();
void Inorder();
void Preorder();
struct node
{
    int data;
    struct node* left;
    struct node* right;
};
struct node* newNode(int data)
{
    struct node* node = (struct node*)
      malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}
void Postorder(struct node* node) {
    if (node == NULL)
      return;
    Postorder(node->left);
    Postorder(node->right);
    printf("%d ", node->data);
}
void Inorder(struct node* node) {
    if (node == NULL)
        return;
    Inorder(node->left);
    printf("%d ", node->data);
    Inorder(node->right);
}
void Preorder(struct node* node) {
    if (node == NULL)
        return;
    printf("%d ", node->data);
    Preorder(node->left);
    Preorder(node->right);
}
void main()
{
    struct node *root  = newNode(29);
    root->left           = newNode(4);
    root->left->left     = newNode(18);
    root->left->right    = newNode(16);
```

```
    root->right          = newNode(2);
    root->right->left    = newNode(7);
    root->right->right   = newNode(9);

   printf("\nPreorder traversal of binary tree is \n");
   Preorder(root);
   printf("\nInorder traversal of binary tree is \n");
   Inorder(root);
   printf("\n Postorder traversal of binary tree is \n");
   Postorder(root);

}
```

Output:

```
Preorder traversal of binary tree is
29 4 18 16 2 7 9
Inorder traversal of binary tree is
18 4 16 29 7 2 9
Postorder traversal of binary tree is
18 16 4 7 9 2 29
```

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

code:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
  int data;
  struct node *left;
  struct node *right;
} node;

node *create()
{
   node *p;
   int x;
   printf("Enter data(-1 for no node):");
   scanf("%d",&x);

   if(x==-1)
         return NULL;

   p=(node*)malloc(sizeof(node));
```

```
    p->data=x;
    printf("Enter left child of %d:\n",x);
    p->left=create();
    printf("Enter right child of %d:\n",x);
    p->right=create();
    return p;
}

void inorder(node *t)
{
  if(t!=NULL)
  {
    inorder(t->left);
    printf("  %d",t->data);
    inorder(t->right);
  }
}


void main()
{
  node *root;
  root=create();

  printf("\nThe inorder traversal of tree is: ");
  inorder(root);

}
```

Output:

```
Enter data(-1 for no node):34
Enter left child of 34:
Enter data(-1 for no node):54
Enter left child of 54:
Enter data(-1 for no node):90
Enter left child of 90:
Enter data(-1 for no node):-1
Enter right child of 90:
Enter data(-1 for no node):-1
Enter right child of 54:
Enter data(-1 for no node):-1
```

```
Enter left child of 34:
Enter data(-1 for no node):
```

3.Write a C program for linear search algorithm.

Code:

```
#include <stdio.h>
void main()
{
  int arr[50], search, 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", &arr[i]);

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

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

Output:

```
Enter number of elements in array
3
Enter 3 integer(s)
1
4
3
Enter a number to search
```

```
3
3 is present at location 3.
```

4.Write a C program for binary search algorithm.

Code:

```
#include <stdio.h>

int main()
{
  int i, first, mid, last, n, search, arr[10];
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (i = 0; i < n; i++){
    scanf("%d", &arr[i]);
  printf("Enter value to find\n");
  scanf("%d", &search);}
  first = 0;
  last = n - 1;
  mid = (first+last)/2;
  while (first <= last) {
    if (arr[mid] < search)
      first = mid + 1;
    else if (arr[mid] == search) {
      printf("%d found at location %d.\n", search, mid+1);
      break;
    }
    else
      last = mid - 1;
    mid = (first + last)/2;
  }
  if (first > last){
  printf("Not present: %d isn't present in the list.\n", search);}
  }
```

Output:

```
Enter number of elements
3
Enter 3 integers
1
2
3
Enter value to find
1
1 found at location 1.
```