1)Write a c program to print preorder,inorder and post order transversal on binary tree.

```
#include <stdio.h>
#include <stdlib.h>
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 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  = newNode(3);
    root->left        = newNode(5);
    root->right       = newNode(6);
    root->left->left  = newNode(7);
    root->left->right  = newNode(9);

    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);

    getchar();
    return 0;
}
```

2)Write a c program to create and inorder transversal on Binary search tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node* left;
struct node* right;
};
struct node* createNode(value){
struct node* newNode = malloc(sizeof(struct node));
newNode->data = value;
newNode->left = NULL;
newNode->right = NULL;
return newNode;
}
struct node* insert(struct node* root, int data)
{
if (root == NULL) return createNode(data);

if (data < root->data)
root->left = insert(root->left, data);
else if (data > root->data)
root->right = insert(root->right, data);
return root;
}
void inorder(struct node* root){
if(root == NULL) return;
inorder(root->left);
printf("%d ->", root->data);
inorder(root->right);
}
int main(){
struct node *root = NULL;
root = insert(root, 20);
insert(root, 10);
insert(root, 30);
insert(root, 40);
insert(root, 50);

insert(root, 60);
```

```
insert(root, 70);
insert(root, 80);
inorder(root);
}
```

3)Write a c program depth-first search using the array.

```
#include<stdio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
 int i;
 reach[v]=1;
 for(i=1;i<=n;i++)
  if(a[v][i] && !reach[i])
  {
   printf("n %d->%d",v,i);
   dfs(i);
  }
}
int main()
{
 int i,j,count=0;
 printf("n Enter number of vertices:");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  reach[i]=0;
  for(j=1;j<=n;j++)
   a[i][j]=0;
 }
 printf("n Enter the adjacency matrix:n");
 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
   scanf("%d",&a[i][j]);
 dfs(1);
 printf("n");
 for(i=1;i<=n;i++)
 {
  if(reach[i])
   count++;
 }
 if(count==n)
  printf("n Graph is connected");
```

```
 else
  printf("n Graph is not connected");
 return 0;
}
```

4)Write a c program berth first search using array.

```
#include<stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v) {
 for(i = 1; i <= n; i++)
 if(a[v][i] && !visited[i])
 q[++r] = i;
 if(f <= r) {
 visited[q[f]] = 1;
 bfs(q[f++]);
 }
}

void main() {
 int v;
 printf("\n Enter the number of vertices:");
 scanf("%d", &n);

 for(i=1; i <= n; i++) {
 q[i] = 0;
 visited[i] = 0;
 }

 printf("\n Enter graph data in matrix form:\n");
 for(i=1; i<=n; i++) {
 for(j=1;j<=n;j++) {
 scanf("%d", &a[i][j]);
 }
 }

 printf("\n Enter the starting vertex:");
 scanf("%d", &v);
 bfs(v);
 printf("\n The node which are reachable are:\n");
```

```
 for(i=1; i <= n; i++) {
 if(visited[i])
 printf("%d\t", i);
 else {
 printf("\n Bfs is not possible. Not all nodes are reachable");
 break;
 }
 }
}
```

5)Write a 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)
    {
      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;
}
```

6)Write a c program for a 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;
}
```