Name: R.Ramya Manasa
Id no: AP19110010472
Sec: CSE-F

LAB PROGRAMS

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;
};
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(4);
    root->left             = newNode(6);
    root->right           = newNode(8);
    root->left->left     = newNode(10);
    root->left->right   = newNode(12);

    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 (or insert) and inorder traversal on Binary Search Tree.

Code:

```
#include<stdio.h>
#include<conio.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);

  getch();
}
```

3)Write a C program depth first search (DFS) using array.

Code:

```
#include<stdio.h>

void DFS(int);
int G[10][10],visited[10],n;

void main()
{
    int i,j;
    printf("Enter number of vertices:");

            scanf("%d",&n);

            printf("\nEnter adjecency matrix of the graph:");

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

   for(i=0;i<n;i++)
        visited[i]=0;

    DFS(0);
}

void DFS(int i)
{
    int j;
        printf("\n%d",i);
    visited[i]=1;

        for(j=0;j<n;j++)
      if(!visited[j]&&G[i][j]==1)
        DFS(j);
}
```

4)Write a C program breath first search (BFS) 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");
           }
    }
}
```

5)Write a C program for linear search algorithm.

Code:

```
#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;
}
```

6)Write a C program for binary search algorithm.

Code:

```
#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;
}
```