AP19110010380
CSE-H

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

```
// C program for different tree traversals
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
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);
}

/* Driver program to test above functions*/
int main()
{
     struct node *root  = newNode(1);
     root->left             = newNode(2);
     root->right           = newNode(3);
     root->left->left     = newNode(4);
```

```
    root->left->right   = newNode(5);

    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;
}
```

OUTPUT:

```
Preorder transversal of binary tree is

1 2 4 5 3

Inorder transversal of binary tree is

4 2 5 1 3

Postorder transversal of binary tree is

4 5 2 3 1
```

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

```
// C program to insert operation in binary search tree.
#include<stdio.h>
#include<stdlib.h>
```

```
struct node
{
    int key;
    struct node *left, *right;
};

// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to do inorder traversal of BST
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d \n", root->key);
        inorder(root->right);
    }
}
struct node* insert(struct node* node, int key)
{
    if (node == NULL) return newNode(key);
    if (key < node->key)
```

```
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    return node;
}

// Driver Program to test above functions
int main()
{
    /* Let us create following BST
              50
           /     \
          30      70
         /  \    /  \
       20   40  60   80 */
    struct node *root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);

    // print inoder traversal of the BST
    inorder(root);

    return 0;
}
```

OUTPUT:

```
20

30

40

50

60

70

80
```

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

```
// C program depth first search (DFS) using array.

#include <stdio.h>
#include <stdlib.h>
int source,V,E,time,visited[20],G[20][20];
void DFS(int i)
{
    int j;
    visited[i]=1;
    printf(" %d->",i+1);
    for(j=0;j<V;j++)
    {
        if(G[i][j]==1&&visited[j]==0)
            DFS(j);
    }
}
int main()
```

```
{
   int i,j,v1,v2;
   printf("\t\t\tGraphs\n");
   printf("Enter the no of edges:");
   scanf("%d",&E);
   printf("Enter the no of vertices:");
   scanf("%d",&V);
   for(i=0;i<V;i++)
   {
      for(j=0;j<V;j++)
         G[i][j]=0;
   }
   /*    creating edges :P    */
   for(i=0;i<E;i++)
   {
      printf("Enter the edges (format: V1 V2) : ");
      scanf("%d%d",&v1,&v2);
      G[v1-1][v2-1]=1;

   }

   for(i=0;i<V;i++)
   {
      for(j=0;j<V;j++)
         printf(" %d ",G[i][j]);
      printf("\n");
   }
   printf("Enter the source: ");
   scanf("%d",&source);
      DFS(source-1);
```

```
    return 0;
}
```

OUTPUT:

```
Enter the number of vertices: 8

Enter adjacency matrix of graph: 0 1 1 1 1 0 0 0

1 0 0 0 0 1 0 0

1 0 0 0 0 1 0 0

1 0 0 0 0 0 1 0

1 0 0 0 0 0 1 0

0 1 1 0 0 0 0 1

0 0 0 1 1 0 0 1

0 0 0 0 0 0 1 1


0

1

5

2

7

6

3

4
```

4.Write a C program breath first search (BFS) using array.

```
#include<stdio.h>
int G[20][20],q[20],visited[20],n,front = 1, rear = 0 ;
void bfs(int v)
```

```
{
    int i;
    visited[v] = 1;
 for(i=1;i<=n;i++)
  if(G[v][i] && !visited[i])
   q[++rear]=i;
   if(front <= rear)
    bfs(q[front++]);
 }

int main()
{
 int v,i,j;

 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",&G[i][j]);
 printf("\n Enter the starting vertex:");
 scanf("%d",&v);
 bfs(v);
 printf("\n The nodes which are reachable are:\n");
 for(i=1;i<=n;i++)
```

```
  if(visited[i])
   printf("%d\t",i);
  else
   printf("\n %d is not reachable",i);

return 0;
}
```

OUTPUT:

```
Enter the number of vertices:4

Enter the graph in the form of matrix:

1 1 1 1

0 1 0 0

0 0 1 0

0 0 0 1

Enter the starting vertex:1

The node which are reachable are:

      1     2   3   4
```

5. Write a C program for linear search algorithm.

```
// C code to linearly search x in arr[].
#include <stdio.h>

int search(int arr[], int n, int x)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

int main(void)
{
    int arr[] = { 2, 8, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = search(arr, n, x);
    (result == -1)   printf("Element is not present in array")
                   : printf("Element is present at index %d",result);
    return 0;
}
```

OUTPUT:

```
The element is present at index 3.
```

6.Write a C program for binary search algorithm.

```
#include<stdio.h>

int main()
{
   int arr[50],i,n,x,flag=0,first,last,mid;

   printf("Enter size of array:");
   scanf("%d",&n);
   printf("\nEnter array element(ascending order)\n");

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

```
    printf("\nEnter the element to search:");
    scanf("%d",&x);

    first=0;
    last=n-1;

    while(first<=last)
    {
        mid=(first+last)/2;

        if(x==arr[mid]){
            flag=1;
            break;
        }
        else
            if(x>arr[mid])
                first=mid+1;
            else
                last=mid-1;
    }

    if(flag==1)
        printf("\nElement found at position %d",mid+1);
    else
        printf("\nElement not found");

    return 0;
}
```

OUTPUT:

```
Enter the size of array:6

Enter the elements of array: 2 4 6 8 10 12

Enter the element to search: 4

Element found at a position 2
```