Lab Record

Question 1: Write a C program to print pre-order, in-order, and post-order traversal on Binary Tree.

Answer:

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(value)
{
 struct node* newNode = malloc(sizeof(struct node));
 newNode->data = value;
 newNode->left = NULL;
 newNode->right = NULL;
return newNode;
}
struct node* insertL(struct node *root, int value)
 {
 root->left = createNode(value);
 return root->left;
}

struct node* insertR(struct node *root, int value)
```

```
{
 root->right = createNode(value);
 return root->right;
}
void main()
{
 struct node* root = createNode(1000);
 insertL(root, 500);
 insertR(root, 2000);
 insertL(root->left, 200);
 insertR(root->left,800 );
 printf("Inorder traversal is following\n");
 inorder(root);
 printf("\nPreorder traversal is following\n");
 preorder(root);

 printf("\nPostorder traversal is following \n");
 postorder(root);
}
```

Output:

```
Inorder traversal is following
200 500 800 1000 2000
Preorder traversal is following
1000 500 200 800 2000
Postorder traversal is following
200 800 500 2000 1000
```

Question 2: Write a C program to create (or insert) and in-order traversal on Binary Search Tree.

Answer:

Code:

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

struct node
{
int data;
struct node *left, *right;
};

struct node *newNode(int item)
{
struct node *temp = (struct node *)malloc(sizeof(struct node));
temp->data = item;
temp->left = temp->right = NULL;
return temp;
}

void inorder(struct node *root)
{
if (root != NULL)
{
inorder(root->left);
printf("%d \t", root->data);
```

```
inorder(root->right);
}
}

struct node* insert(struct node* node, int data)
{
if (node == NULL)
return newNode(data);
if (data< node->data)
node->left  = insert(node->left, data);
else if (data > node->data)
node->right = insert(node->right, data);
return node;
}

void main()
{
struct node *root = NULL;
root = insert(root, 1000);
insert(root, 500);
insert(root, 200);
insert(root, 800);
insert(root, 2000);
insert(root, 1500);
insert(root, 4000);
printf("Inorder Traversal is following \n");
inorder(root);
```

}

Output:

```
Inorder Traversal is following

200     500     800     1000    1500    2000    4000
```

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

Answer:

Code:

```
#include<stdio.h>
int a[10][10],visited[10],n;

void dfs(int x)
{
int i;
visited[x]=1;
for (i=1;i<=n;i++)
if(a[x][i]==1 && !visited[i])
{
printf("\n %d->%d",x,i);
dfs(i);
}
}

void main()
```

```
{
int i,j,count=0;
printf("\n Enter number of vertices:");
scanf("%d",&n);
for (i=1;i<=n;i++)
{
visited[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(visited[i])
count++;
}
if(count==n)
printf("\n Matrix CONNECTED");
else
printf("\n Matrix NOT CONNECTED");
}
```

Output:

```
 Enter number of vertices:2


 Enter the adjacency matrix:
0
0
0
0


 Matrix NOT CONNECTED
```

Question 4: Write a C program for breadth first search (BFS) using array.

Answer:

Code:

```
#include<stdio.h>
int a[10][10],s[10],visited[10],n,i,j,f=0,r=-1;

void bfs(int x)
{
for (i=1;i<=n;i++)
if(a[x][i] && !visited[i])
s[++r]=i;
if(f<=r)
{
```

```
visited[s[f]]=1;
bfs(s[f++]);
}
}

void main()
{
int p;
printf("\n Enter the number of vertices:");
scanf("%d",&n);
for (i=1;i<=n;i++)
{
s[i]=0;
visited[i]=0;
}
printf("\n Enter the 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",&p);
bfs(p);
printf("\n The node that are reachable:\n");
```

```
for (i=1;i<=n;i++)
{
if(visited[i])
printf("%d\t",i);
else
printf("\n BFS isn't possible");
}
}
```

Output:

```
Enter the number of vertices:2


 Enter the data in matrix form:
1
1
0
0

 Enter the starting vertex:1


 The node that are reachable:
1       2
```

Question 5: Write a C program for linear search algorithm.

Answer:

Code:

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

```
int search(int a[5], int p)
{
int i;
for (i=0; i<5; i++)
{
if (a[i]==p)
return i;
}
return -1;
}

void main()
{
int p,i,a[5];
printf("Enter five numbers in the array \n");
for(i=0;i<5;i++)
{
scanf("%d\n",&a[i]);
}
printf("Enter the number to be searched \t");
scanf("%d",&p);
int result = search(a,p);
if(result==-1)
printf("Element is not present in array");
else
printf("Element is present at position %d", result+1);
}
```

Output:

```
Enter five numbers in the array
1
2
3
4
5
Enter the number to be searched  4
Element is present at position  4
```

Question 6: Write a C program for binary search algorithm

Answer:

Code:

```
#include <stdio.h>
int binarySearch(int a[], int m, int r, int x)
{
if (r>= m)
{
int mid = m + (r-m)/2;
if (a[mid]==x)
return mid;
else if (a[mid]>x)
return binarySearch(a, m, mid - 1, x);
return binarySearch(a, mid + 1, r, x);
}
return -1;
}
```

```
void main()
{
int p,i,a[5];
printf("Enter five numbers in the array \n");
for(i=0;i<5;i++)
{
scanf("%d\n",&a[i]);
}
printf("Enter the number to be searched \t");
scanf("%d",&p);
int result = binarySearch(a, 0, 4, p);
if(result==-1)
printf("Element is not present in array");
else
printf("Element is present at position %d", result+1);
}
```

Output:

```
Enter five numbers in the array
1
2
3
4
5
Enter the number to be searched  4
Element is present at position  4
```

****