Lab Exercises:

1). Write a program to determine the Topological sort of a given graph using
i. Depth-First technique
ii. Source removal technique

Program (i):

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

int a[50][50], visit[50], stack[100],n,t=0;

void dfs(int v)
{
	visit[v]=1;

	for(int i=0;i<n;i++)
	{
		if(a[v][i] && !visit[i])
		{
			dfs(i);
		}
	}

	stack[t++]=v;
}

void printStack()
{
	for(int i=n-1;i>=0;i--)
	{
		printf("%d\n",stack[i]);
	}

	printf("\n");
}

int main()
{
	printf("Enter the Number of Vertices : \n");
	scanf("%d", &n);

	printf("Enter the Adjacency Matrix : \n");

	for(int i = 0; i<n; i++)
```

```
        {
                for(int j = 0; j<n; j++)
                {
                scanf("%d", &a[i][j]);
                }
        }

        for(int i = 0; i<n; i++)
        {
                if(!visit[i])
                {
                dfs(i);
                }
        }

        printf("The Topological Sort Order is :\n");
        printStack();


        return 0;
}
```

Output:

```
Enter the Number of Vertices :
5
Enter the Adjacency Matrix :
0 0 1 0 0
0 0 1 0 0
0 0 0 1 1
0 0 0 1 0
0 0 0 0 0
The Topological Sort Order is :
1
0
2
4
3


Process returned 0 (0x0)   execution time : 151.333 s
Press any key to continue.
```

Program (ii):

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

int queue[100], k_1 = 0, k = 0, arr[100][100], n, indegree[100];
void calc()
{
	for(int i = 0; i<n; i++)
	{
		for(int j = 0; j<n; j++)
		{
			if(arr[j][i] && i!=j)
			{
			indegree[i]++;
			}
		}
	}
}

void initQueue()
{
	for(int i = 0; i<n; i++)
	{
	queue[i] = -1;
	}
}

void dec(int v)
{
	for(int i = 0; i<n; i++)
	{
		if(arr[v][i])
		{
			indegree[i]--;
		}

		if(indegree[i] == 0)
		{
		queue[k++] = i;
		}
	}
}

int queueEmpty()
{
	for(int i = 0; i<n; i++)
	{
		if(queue[i] != -1)
		{
		return 0;
```

```
                }
        }

        return 1;
}

int main()
{
        printf("Enter the Number of Vertices : \n");
        scanf("%d", &n);

        printf("Enter the Adjacency Matrix : \n");

        for(int i = 0; i<n; i++)
        {
                for(int j = 0; j<n; j++)
                {
                scanf("%d", &arr[i][j]);
                }
        }

        initQueue();
        calc();

        for(int i = 0; i<n; i++)
        {
                if(indegree[i] == 0)
                {
                queue[k++] = i;
                }
        }

        printf("The Topological Sort Order is : \n");

        while(!queueEmpty())
        {
                int vertex = queue[k_1++];
                printf("%d ", vertex);
                queue[k_1-1] = -1;
                dec(vertex);

                printf("%d ", vertex);
        }

        printf("\n");

        return 0;
}
```

Output:

```
Enter the Number of Vertices :
3
Enter the Adjacency Matrix :
0 1 1
0 0 1
0 0 0
The Topological Sort Order is :
0
1
2


Process returned 0 (0x0)   execution time : 22.237 s
Press any key to continue.
```

2. Write a program to find diameter of a binary tree. Diameter of a binary tree is the longest path between any two nodes.

Program:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int val;
        struct node *left, *right;
};
struct node* newNode(int value)
{
        struct node* node
                = (struct node*)malloc(sizeof(struct node));
        node->val = value;
        node->left = NULL;
        node->right = NULL;
        return (node);
}
int max(int a, int b)
{
   return (a > b) ? a : b;
}
int height(struct node* node)
{
        if (node == NULL)
                return 0;
        return 1 + max(height(node->left), height(node->right));
}
int diameter(struct node* tree)
{
        if (tree == NULL)
                return 0;
```

```
        int lheight = height(tree->left);
        int rheight = height(tree->right);
        int ldiam = diameter(tree->left);
        int rdiam = diameter(tree->right);
        return max(lheight + rheight + 1, max(ldiam, rdiam));
}
int main()
{
        struct node* root = newNode(1);
        root->left = newNode(2);
        root->left->left = newNode(4);
        root->left->right = newNode(5);
        root->left->right->left= newNode(6);
        root->left->right->right = newNode(7);

        root->right = newNode(3);
        root->right->right = newNode(8);
        root->right->right->right = newNode(9);
        root->right->right->right->right = newNode(10);
        root->right->right->right->left= newNode(11);
        root->right->right->right->left->left = newNode(12);
        root->right->right->right->left->right = newNode(13);
        root->right->right->right->left->right->left = newNode(14);
        root->right->right->right->left->right->right = newNode(15);
        printf("Diameter of the given binary tree is %d\n",
                diameter(root));
        return 0;
}
```

Output:

```
Diameter of the given binary tree is 10

Process returned 0 (0x0)   execution time : 1.391 s
Press any key to continue.
```