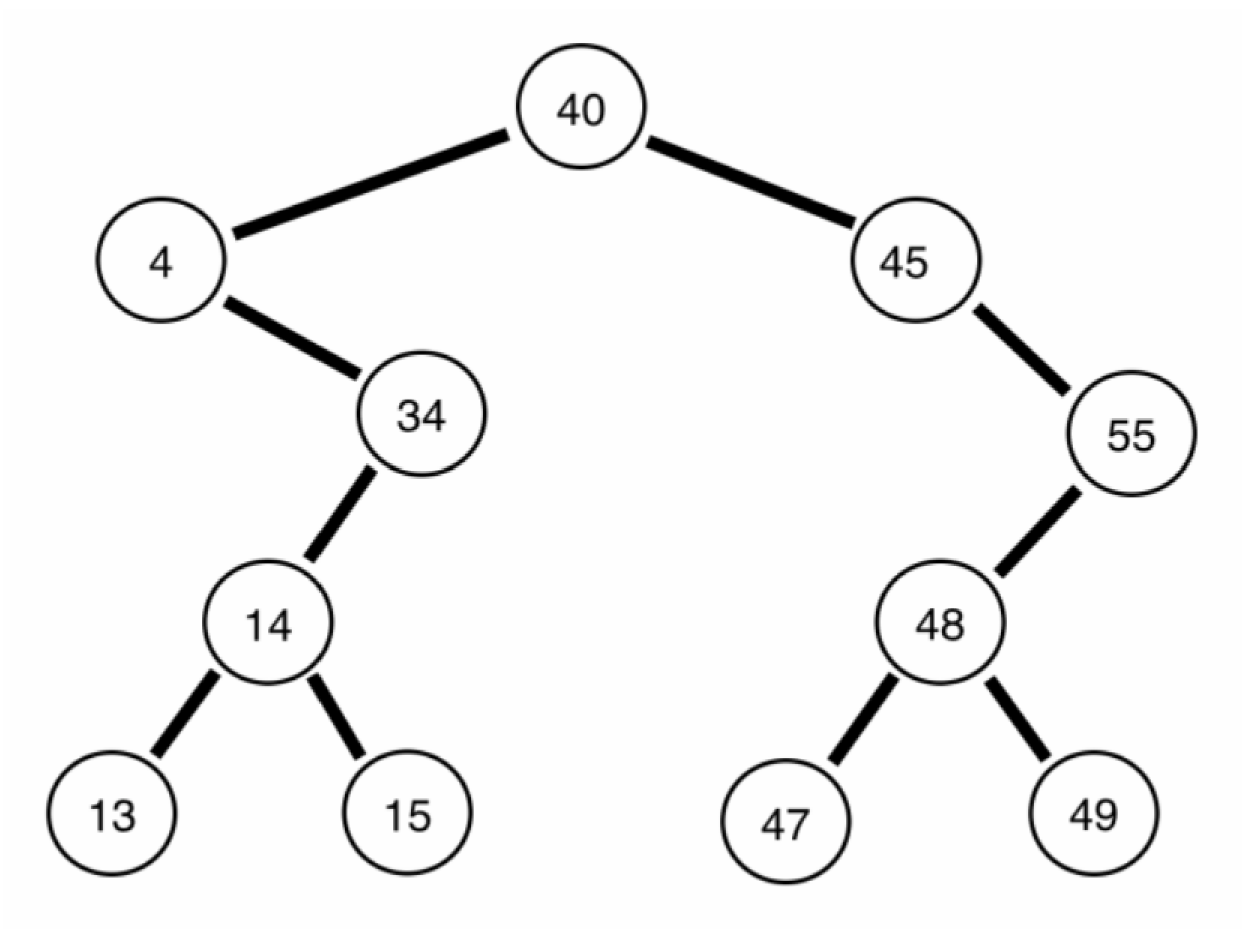
Final Delivery

Part A

Exercise 1

For a given tree, provide the order of the traversing operation when using:



Types of orders:

- Pre-Order:

```
40 4 34 14 13 15 45 55 48 47 49
```

- In-Order:

```
13 14 15 34 4 47 48 49 55 45 40
```

- Post-Order:

```
13 15 14 34 4 47 49 48 55 45 40
```

Part B

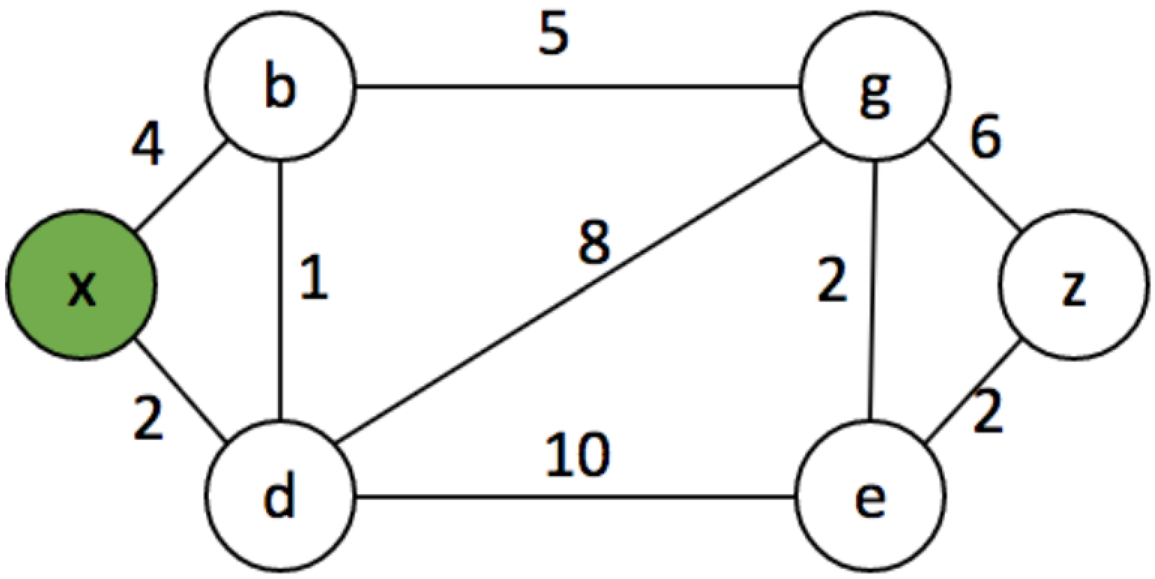
Exercise 1

Draw a corresponding graph for the adjacency matrix.

Exercise 2

For the following bidirected graph:

- Find the adjacency matrix.
- Find the adjacency linked list.
- Go through it using both, BFS and DFS, algorithms. Assume that you start at node x. Indicate the order of visited nodes.



Adjacency Matrix:

```
[   x, b, g, z, d, e]
[x,  , 4,  ,  , 2,  ]
[b, 4,  , 5,  , 1,  ]
[g,  , 5,  , 6, 8, 2]
[z,  ,  , 6,  ,  , 2]
[d, 2, 1, 8,  ,  ,10]
[e,  ,  , 2, 2,10,  ]
```

Adjacency List:

```
x: (b, 4) -> (d, 2)
b: (x, 5) -> (g, 5) -> (d, 1)
g: (b, 5) -> (z, 6) -> (d, 8) -> (3, 2)
z: (g, 6) -> (e, 2)
d: (x, 2) -> (b, 1) -> (g, 8) -> (3, 10)
e: (g, 2) -> (z, 2) -> (d, 10)
```

BFS Algorithms Traversal:

```
x -> b -> d -> g -> e -> z
```

DFS Algorithms Traversal:

```
x -> b -> g -> z -> d -> e
```

Exercise 3

Code of the program `graph.c`:

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

struct Graph{
    int numNodes;
    int ** matrix;
};

void add_node(int from, int to, int weight, struct Graph graph, char binar)
```

```c
{
    graph.numNodes++;
    graph.matrix[from][to] = weight;
    if (binar == 'y')
        graph.matrix[to][from] = weight;
}

void print_graph(struct Graph graph, int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            printf("%d ",graph.matrix[i][j]);
        printf("\n");
    }
}

int main(){
    // Create the graph
    int n;
    char binar;
    //########
    n = 5;
    binar = 'y';
    //########
    struct Graph graph;
    graph.matrix = malloc(sizeof(int*)*n);
    for(int i=0;i<n;i++)
    {
        graph.matrix[i]= malloc(sizeof(int)*n);
    }

    add_node(0, 1, 5, graph, binar);
    add_node(0, 4, 3, graph, binar);
    add_node(1, 2, 2, graph, binar);
    add_node(1, 3, 7, graph, binar);
    add_node(1, 4, 6, graph, binar);
    add_node(3, 4, 5, graph, binar);

    print_graph(graph, n);
    for(int i=0;i<n;i++)
    {
        free(graph.matrix[i]);
    }
    free(graph.matrix);
    return 0;
}
```