Lab 5 : Decrease and Conquer : Dipesh Singh 190905520

Question 1 : Topological Sorting using DFS

```
```

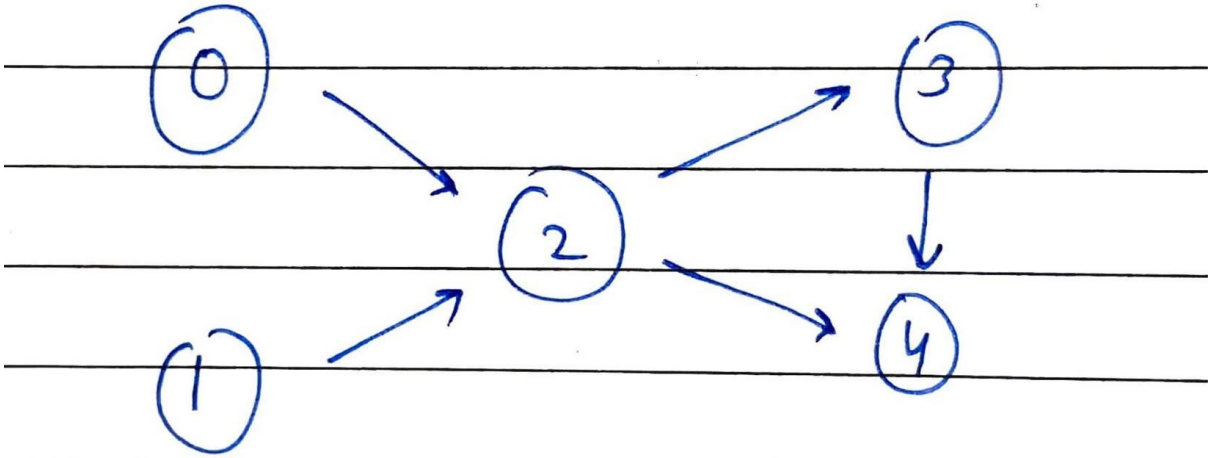
```c
#include <stdio.h>
#include <stdlib.h>

int isEmpty(int top)
{
    if (top == -1)
    {
        return 1;
    }
    return 0;
}

int isFull(int top, int capacity)
{
    if (top == capacity - 1)
    {
        return 1;
    }
    return 0;
}

void push(int **Stack, int *top, int *capacity, int key)
{
    if (isFull(*top, *capacity))
    {
        *Stack = (int *)realloc(*Stack, sizeof(int) * (*capacity) * 2);
        *capacity *= 2;
    }
    (*top)++;
    (*Stack)[*top] = key;
}

int pop(int **Stack, int *top)
{
    int temp = (*Stack)[*top];
    (*top)--;
    return temp;
}

void display(int *Stack, int top)
{
    if (isEmpty(top))
    {
    }
```

```
    else
    {
        printf("stack : ");
        int i;
        for (i = 0; i <= top; i++)
        {
            printf("%d ", *(Stack + i));
        }
        printf("\n");
    }
}

void insertEdgeM(int **matrix, int first, int second)
{
    matrix[first][second] = 1;
}

void displayMatrix(int **matrix, int n)
{
    for (int i = -1; i < n; i++)
    {
        if (i != -1)
            printf("%d -> ", i);
        for (int j = 0; j < n; j++)
        {
            if (i == -1)
            {
                if (j == 0)
                {
                    printf("\t");
                }
                printf("%d\t", j);
                continue;
            }
            if (j == 0)
            {
                printf("\t");
            }
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

void dfs(int **matrix, int num, int **Stack, int *top, int *capacity)
{
    int *visited = (int *)calloc(num, sizeof(int));
    for (int i = 0; i < num; i++)
```

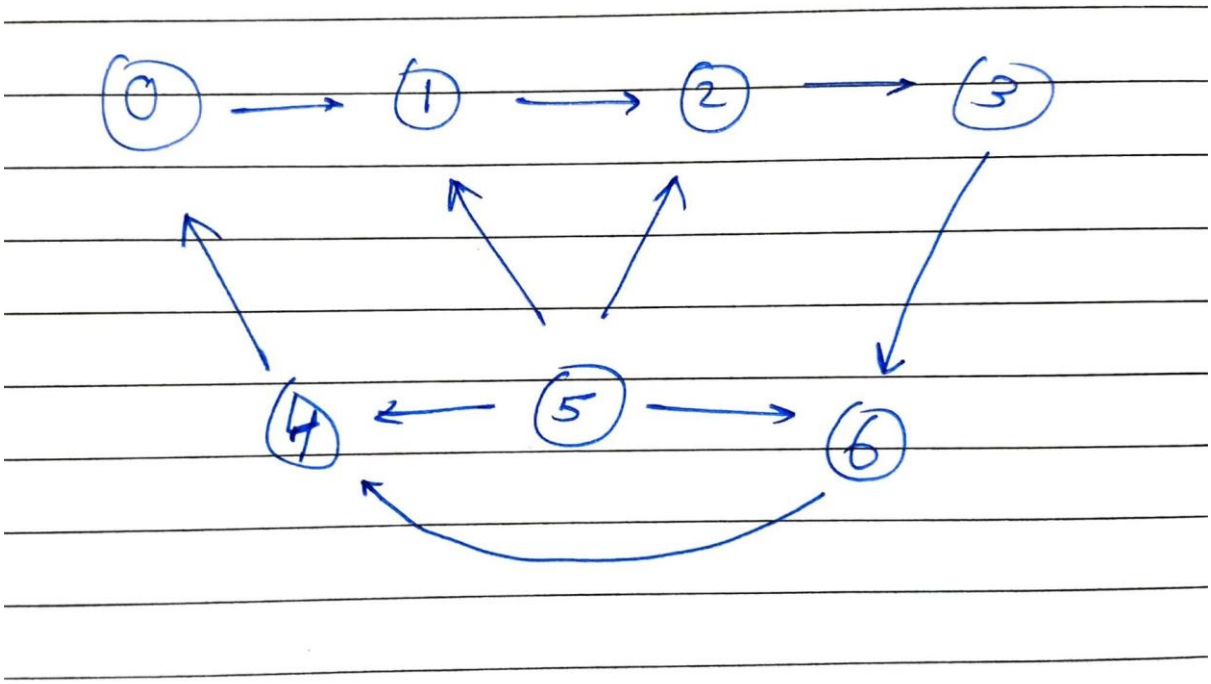
```
    {
        visited[i] = 0;
    }
    char result[100];
    int resultIndex = 0;
    char popped[100];
    int poppedIndex = 0;
    push(Stack, top, capacity, 0);
    printf("pushed : %d\n", 0);
    char p = (char)('0' + 0);
    result[resultIndex++] = p;
    display(*Stack, *top);
    visited[0] = 1;
    int cur = *Stack[*top];
    int flag, ele;
    while (1)
    {
        if (!(isEmpty(*top)))
        {
            flag = 0;
            for (int i = 0; i < num; i++)
            {
                if (visited[i] == 0 && matrix[cur][i] == 1)
                {
                    visited[i] = 1;
                    printf("pushed : %d\n", i);
                    p = (char)('0' + i);
                    result[resultIndex++] = p;
                    push(Stack, top, capacity, i);
                    display(*Stack, *top);
                    flag = 1;
                    break;
                }
            }
            if (flag == 0)
            {
                ele = pop(Stack, top);
                p = (char)('0' + ele);
                popped[poppedIndex++] = p;
                printf("popped : %d\n", ele);
                display(*Stack, *top);
            }
            cur = (*Stack)[*top];
        }
        else
        {
            flag = 1;
            for (int i = 0; i < num && flag; i++)
```

```
            {
                if (visited[i] == 0)
                {
                    visited[i] = 1;
                    printf("pushed : %d\n", i);
                    p = (char)('0' + i);
                    result[resultIndex++] = p;
                    push(Stack, top, capacity, i);
                    display(*Stack, *top);
                    flag = 0;
                    cur = (*Stack)[*top];
                    break;
                }
            }
            if (flag == 1)
            {
                break;
            }
        }
    }
    while (!(isEmpty(*top)))
    {
        int rem = pop(Stack, top);
        p = (char)('0' + rem);
        popped[poppedIndex++] = p;
        printf("popped : %d\n", rem);
        display(*Stack, *top);
    }
    printf("The dfs is : ");
    for (int i = 0; i < resultIndex; i++)
    {
        printf("%c ", result[i]);
    }
    printf("\nThe traversal stack is : ");
    for (int i = 0; i < poppedIndex; i++)
    {
        printf("%c ", popped[i]);
    }
    printf("\n");
    for (int i = 0; i < poppedIndex - 1; i++)
    {
        for (int j = i + 1; j < poppedIndex; j++)
        {
            if (matrix[((int)(popped[i] - '0'))][((int)(popped[j] - '0'))
] == 1)
            {
                printf("The graph is not a Directed Acyclic Graph, topolo
gical sort not possible.\n");
```

```
                return;
            }
        }
    }
    printf("The graph is a Directed Acyclic Graph, topoplogical sort is p
ossible : ");
    for (int i = poppedIndex - 1; i >= 0; i--)
    {
        printf("%c ", popped[i]);
    }
    printf("\n");
}

int main()
{
    int num = 5;
    int **matrix = (int **)calloc(num, sizeof(int *));
    for (int i = 0; i < num; i++)
    {
        matrix[i] = (int *)calloc(num, sizeof(int));
        for (int j = 0; j < num; j++)
        {
            matrix[i][j] = 0;
        }
    }
    insertEdgeM(matrix, 0, 2);
    insertEdgeM(matrix, 1, 2);
    insertEdgeM(matrix, 2, 3);
    insertEdgeM(matrix, 2, 4);
    insertEdgeM(matrix, 3, 4);
    displayMatrix(matrix, num);
    int top = -1, capacity = 2;
    int *Stack = (int *)calloc(capacity, sizeof(int));
    dfs(matrix, num, &Stack, &top, &capacity);
    return 0;
}
```




```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ make topoDfs
cc     topoDfs.c   -o topoDfs
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./topoDfs
        0       1       2       3       4
0 ->    0       0       1       0       0
1 ->    0       0       1       0       0
2 ->    0       0       0       1       1
3 ->    0       0       0       0       1
4 ->    0       0       0       0       0
pushed : 0
stack : 0
pushed : 2
stack : 0 2
pushed : 3
stack : 0 2 3
pushed : 4
stack : 0 2 3 4
popped : 4
stack : 0 2 3
popped : 3
stack : 0 2
popped : 2
stack : 0
popped : 0
pushed : 1
stack : 1
popped : 1
The dfs is : 0 2 3 4 1
The traversal stack is : 4 3 2 0 1
The graph is a Directed Acyclic Graph, topoplogical sort is possible : 1 0 2 3 4
```




```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./topoDfs
        0       1       2       3       4       5       6
0 ->    0       1       0       0       0       0       0
1 ->    0       0       1       0       0       0       0
2 ->    0       0       0       1       0       0       0
3 ->    0       0       0       0       0       0       1
4 ->    1       0       0       0       0       0       0
5 ->    0       1       1       0       1       0       1
6 ->    0       0       0       0       1       0       0
pushed : 0
stack : 0
pushed : 1
stack : 0 1
pushed : 2
stack : 0 1 2
pushed : 3
stack : 0 1 2 3
pushed : 6
stack : 0 1 2 3 6
pushed : 4
stack : 0 1 2 3 6 4
popped : 4
stack : 0 1 2 3 6
popped : 6
stack : 0 1 2 3
popped : 3
stack : 0 1 2
popped : 2
stack : 0 1
popped : 1
stack : 0
popped : 0
pushed : 5
stack : 5
popped : 5
The dfs is : 0 1 2 3 6 4 5
The traversal stack is : 4 6 3 2 1 0 5
The graph is not a Directed Acyclic Graph, topological sort not possible.
```

The time complexity of DFS using adjacency matrix is $O(V^2)$ hence the time complexity of Topological sorting using DFS is $O(V^2)$.

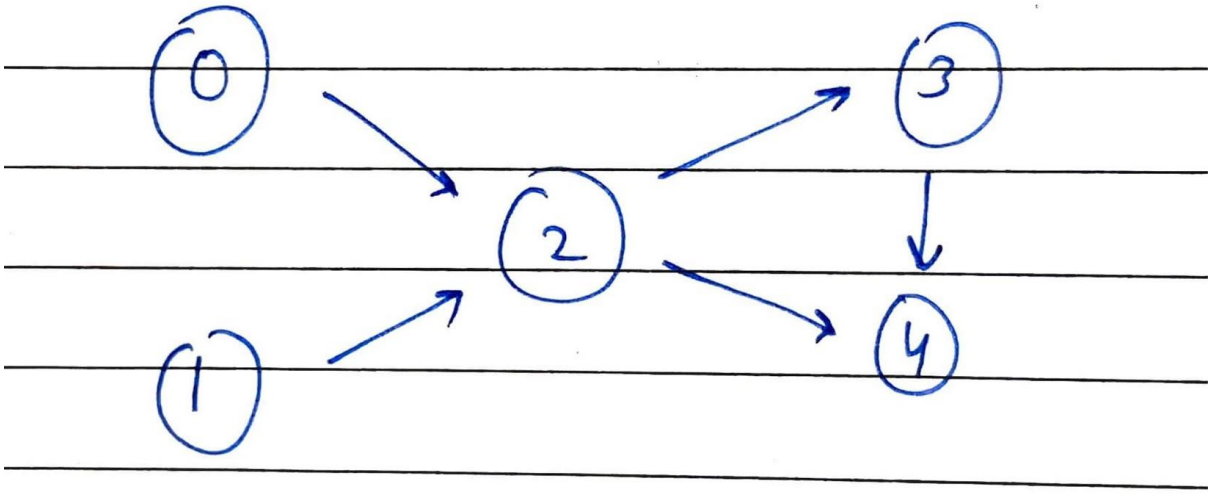
Question 2 : Topological Sorting using Source Removal Technique

```
```

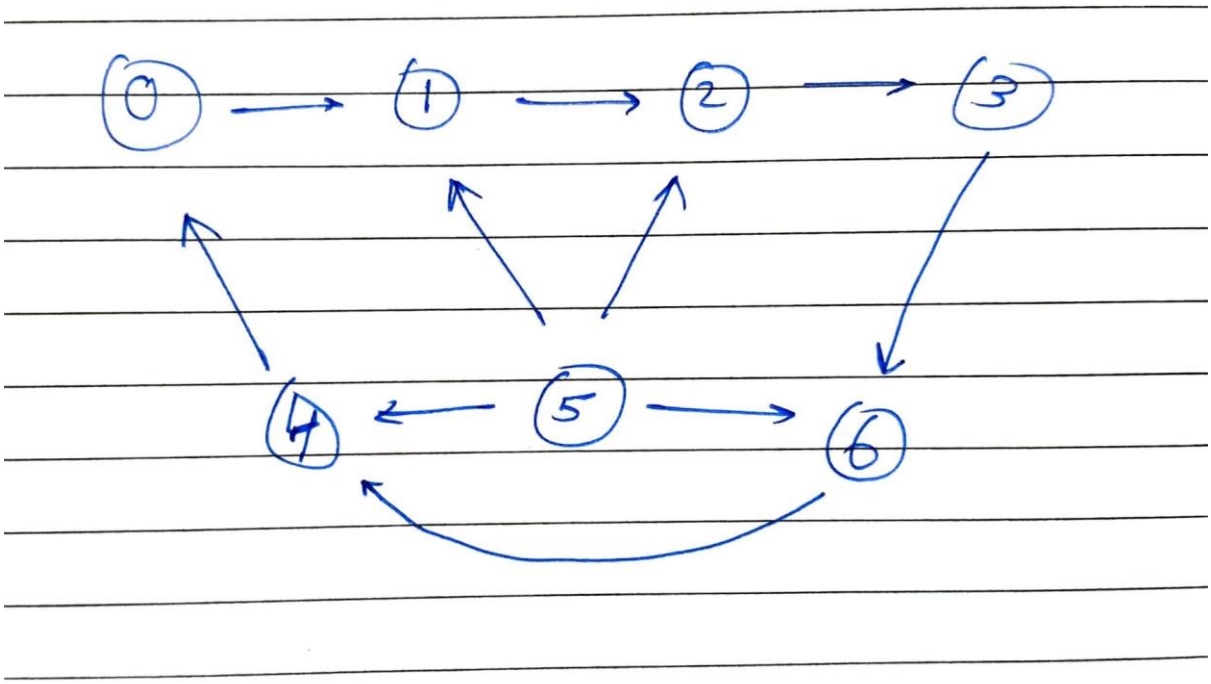
```c
#include <stdio.h>
#include <stdlib.h>

void insertEdgeM(int **matrix, int first, int second)
{
    matrix[first][second] = 1;
}

void displayMatrix(int **matrix, int n)
{
    for (int i = -1; i < n; i++)
    {
        if (i != -1)
            printf("%d -> ", i);
        for (int j = 0; j < n; j++)
        {
            if (i == -1)
            {
                if (j == 0)
                {
                    printf("\t");
                }
                printf("%d\t", j);
                continue;
            }
            if (j == 0)
            {
                printf("\t");
            }
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

int check(int **matrix, int num, int node)
{
    int result = 1;
    for (int i = 0; i < num; i++)
    {
        if (matrix[i][node] == 1)
        {
            result = 0;
        }
    }
```

```
    return result;
}

void deleteEdges(int **matrix, int num, int node)
{
    for (int i = 0; i < num; i++)
    {
        matrix[node][i] = 0;
    }
}

void topo(int **matrix, int num)
{
    int *removed = (int *)calloc(num, sizeof(int));
    for (int i = 0; i < num; i++)
    {
        removed[i] = 0;
    }
    int popped[num];
    int poppedIndex = 0;
    for (int i = 0; i < num; i++)
    {
        if (removed[i] == 0 && check(matrix, num, i))
        {
            removed[i] = 1;
            popped[poppedIndex++] = i;
            deleteEdges(matrix, num, i);
            i = -1;
        }
    }
    for (int i = 0; i < num; i++)
    {
        if (removed[i] == 0)
        {
            printf("The graph is not a Directed Acyclic Graph, topologica
l sort not possible.\n");
            return;
        }
    }
    printf("The graph is a Directed Acyclic Graph, topoplogical sort is p
ossible : ");
    for (int i = 0; i < poppedIndex; i++)
    {
        printf("%d ", popped[i]);
    }
    printf("\n");
}
```

```c
int main()
{
    int num = 5;
    int **matrix = (int **)calloc(num, sizeof(int *));
    for (int i = 0; i < num; i++)
    {
        matrix[i] = (int *)calloc(num, sizeof(int));
        for (int j = 0; j < num; j++)
        {
            matrix[i][j] = 0;
        }
    }
    insertEdgeM(matrix, 0, 2);
    insertEdgeM(matrix, 1, 2);
    insertEdgeM(matrix, 2, 3);
    insertEdgeM(matrix, 2, 4);
    insertEdgeM(matrix, 3, 4);
    displayMatrix(matrix, num);
    topo(matrix, num);
    return 0;
}
```




```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ make topoSrc
cc     topoSrc.c   -o topoSrc
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./topoSrc
        0       1       2       3       4
0 ->    0       0       1       0       0
1 ->    0       0       1       0       0
2 ->    0       0       0       1       1
3 ->    0       0       0       0       1
4 ->    0       0       0       0       0
The graph is a Directed Acyclic Graph, topoplogical sort is possible : 0 1 2 3 4
```



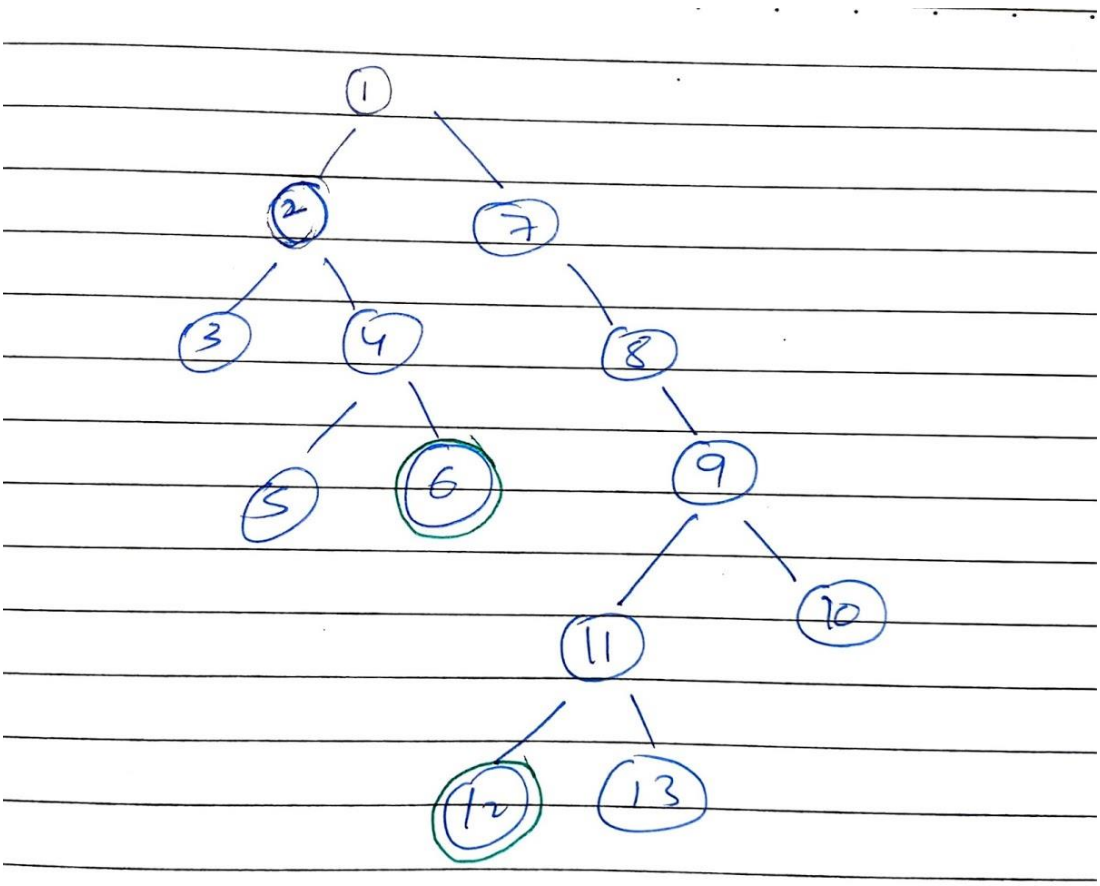

```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ make topoSrc
cc     topoSrc.c   -o topoSrc
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./topoSrc
        0       1       2       3       4       5       6
0 ->    0       1       0       0       0       0       0
1 ->    0       0       1       0       0       0       0
2 ->    0       0       0       1       0       0       0
3 ->    0       0       0       0       0       0       1
4 ->    1       0       0       0       0       0       0
5 ->    0       1       1       0       1       0       1
6 ->    0       0       0       0       1       0       0
The graph is not a Directed Acyclic Graph, topological sort not possible.
```

Since for every vertex V, when deletion operation is underwent, we go through V vertices to remove the edge from the source to the destination, the time complexity of this algorithm is $O(V^2)$.

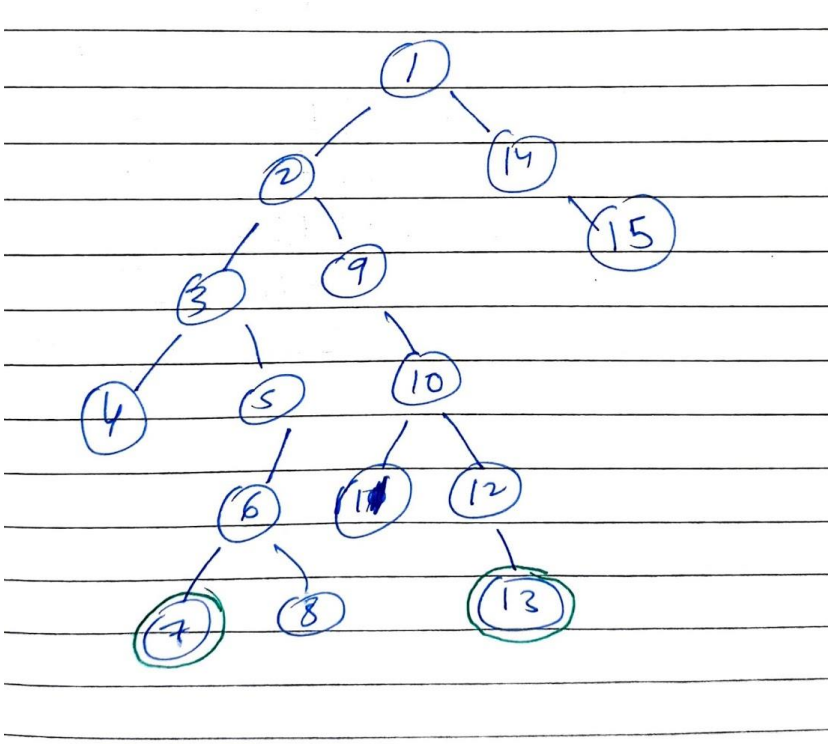
Question 3 : Find the diameter of a tree

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

typedef struct node
{
    int val;
    struct node *left;
    struct node *right;
} * Node;

void inorder(Node n)
{
    if (n)
    {
        inorder(n->left);
        printf("%d ", n->val);
        inorder(n->right);
    }
}

Node insert()
{
    int val;
    int check;
    printf("Enter the element : ");
    scanf("%d", &val);
    Node n = (Node)malloc(sizeof(struct node));
    n->val = val;
    n->left = NULL;
    n->right = NULL;
    printf("Do you want to insert left child of %d? (1 for Yes, 0 for No)
 : ", val);
    scanf("%d", &check);
    if (check)
        n->left = insert();
    printf("Do you want to insert right child of %d? (1 for Yes, 0 for No
) : ", val);
    scanf("%d", &check);
    if (check)
        n->right = insert();
    return n;
}

int max(int a, int b)
```

```
{
    return a > b ? a : b;
}

int height(Node head)
{
    if (head == NULL)
    {
        return 0;
    }
    return max(height(head->left), height(head->right)) + 1;
}

void diameter(Node cur, int *max)
{
    if (cur)
    {
        int curDia = height(cur->left) + height(cur->right) + 1;
        if (curDia > *max)
        {
            *max = curDia;
        }
        diameter(cur->left, max);
        diameter(cur->right, max);
    }
}

int main()
{
    Node head = insert();
    printf("The inorder is : ");
    inorder(head);
    int diam = INT_MIN;
    diameter(head, &diam);
    printf("\nThe diameter of the binary tree is : %d\n", diam);
    return 0;
}
```




```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./diameter
Enter the element : 1
Do you want to insert left child of 1? (1 for Yes, 0 for No) : 1
Enter the element : 2
Do you want to insert left child of 2? (1 for Yes, 0 for No) : 1
Enter the element : 3
Do you want to insert left child of 3? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 3? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 2? (1 for Yes, 0 for No) : 1
Enter the element : 4
Do you want to insert left child of 4? (1 for Yes, 0 for No) : 1
Enter the element : 5
Do you want to insert left child of 5? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 5? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 4? (1 for Yes, 0 for No) : 1
Enter the element : 6
Do you want to insert left child of 6? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 6? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 1? (1 for Yes, 0 for No) : 1
Enter the element : 7
Do you want to insert left child of 7? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 7? (1 for Yes, 0 for No) : 1
Enter the element : 8
Do you want to insert left child of 8? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 8? (1 for Yes, 0 for No) : 1
Enter the element : 9
Do you want to insert left child of 9? (1 for Yes, 0 for No) : 1
Enter the element : 11
Do you want to insert left child of 11? (1 for Yes, 0 for No) : 1
Enter the element : 12
Do you want to insert left child of 12? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 12? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 11? (1 for Yes, 0 for No) : 1
Enter the element : 13
Do you want to insert left child of 13? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 13? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 9? (1 for Yes, 0 for No) : 1
Enter the element : 10
Do you want to insert left child of 10? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 10? (1 for Yes, 0 for No) : 0
The inorder is : 3 2 5 4 6 1 7 8 12 11 13 9 10
The diameter of the binary tree is : 9
```




```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 5$ ./diameter
Enter the element : 1
Do you want to insert left child of 1? (1 for Yes, 0 for No) : 1
Enter the element : 2
Do you want to insert left child of 2? (1 for Yes, 0 for No) : 1
Enter the element : 3
Do you want to insert left child of 3? (1 for Yes, 0 for No) : 1
Enter the element : 4
Do you want to insert left child of 4? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 4? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 3? (1 for Yes, 0 for No) : 1
Enter the element : 5
Do you want to insert left child of 5? (1 for Yes, 0 for No) : 1
Enter the element : 6
Do you want to insert left child of 6? (1 for Yes, 0 for No) : 1
Enter the element : 7
Do you want to insert left child of 7? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 7? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 6? (1 for Yes, 0 for No) : 1
Enter the element : 8
Do you want to insert left child of 8? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 8? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 5? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 2? (1 for Yes, 0 for No) : 1
Enter the element : 9
Do you want to insert left child of 9? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 9? (1 for Yes, 0 for No) : 1
Enter the element : 10
Do you want to insert left child of 10? (1 for Yes, 0 for No) : 1
Enter the element : 11
Do you want to insert left child of 11? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 11? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 10? (1 for Yes, 0 for No) : 1
Enter the element : 12
Do you want to insert left child of 12? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 12? (1 for Yes, 0 for No) : 1
Enter the element : 13
Do you want to insert left child of 13? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 13? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 1? (1 for Yes, 0 for No) : 1
Enter the element : 14
Do you want to insert left child of 14? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 14? (1 for Yes, 0 for No) : 1
Enter the element : 14
Do you want to insert left child of 14? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 14? (1 for Yes, 0 for No) : 1
Enter the element : 15
Do you want to insert left child of 15? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 15? (1 for Yes, 0 for No) : 0
The inorder is : 4 3 7 6 8 5 2 9 11 10 12 13 1 14 15
The diameter of the binary tree is : 9
```