LAB 7

```
tree = {
   'A':['B', 'F', 'D', 'E'],
   'B':['K', 'J'],
   'F':[],
   'D' : ['G'],
   'E' : ['C','H', 'I'],
   'K': ['N', 'M'],
   'J': [],
   'N': [],
   'M': [],
   'G': [],
   'C': [],
   'H': [],
   'I': ['L'],
   'L': []
}


def treedepth(tree, node):
   if not tree[node]:
      return 1
   else:
      return 1 + max(treedepth(tree, child) for child in tree[node])
```

#Depth first search:

```
def dfs(tree, start, target):
    visited = set()
    stack = [start]

    while stack:
        vertex = stack.pop()
        if vertex == target:
            return True
        if vertex not in visited:
            visited.add(vertex)
            stack.extend(reversed(tree[vertex]))
    return False
```

#Depth limit search:

```
def dls(tree, start, target, depth):
    if depth == 0 and start == target:
        return True
    if depth > 0:
        for n in tree[start]:
            if dls(tree, n, target, depth - 1):
                return True

    return False
```

```
# Iterative Deepening Search.
def ids(tree, start, target):
    depth = 0
    td = treedepth(tree, start)
    while True:
        if dls(tree, start, target, depth):
            return True
        if depth > td:
            return False
        depth += 1

print("---------------DFS---------------")
print(dfs(tree, 'A', 'L'))
print(dfs(tree, 'A', 'Z'))
print(dfs(tree, 'A', 'M'))
print("---------------DLS---------------")
print(dls(tree, 'A', 'L', 2))
print(dls(tree, 'A', 'L', 3))
print(dls(tree, 'A', 'L', 4))
print("---------------IDS---------------")
print(ids(tree, 'A', 'L'))
print(ids(tree, 'A', 'Z'))
```

```
print(ids(tree, 'A', 'M'))
```