Group Members:

1. Qui, Christian Michael
2. Regencia, Josiah Eleazar

Requirements:

1. Python2.7.* or Python3

Notes on how to run the homework:

1. Main file is tree_traversal.py
2. Tree_traversal.py accepts an argument to select whether the program will run a dfs or a bfs.
    a. To run bfs: python tree_traversal.py bfs
    b. To run dfs: python tree_traversal.py dfs
3. The arguments 'dfs' or 'bfs' are not case-sensitive. So it can be DFS or Dfs.

Notes on the output:

1. For each node that has been traversed, it will output a line until goal has been reached in the following format:
    - Node: <node address>, is_goal: <Boolean>, node level: <int>
        - We decided to output the node address since there is no context yet. This program was made to be general as possible in preparation of the project.
2. Once goal has been reached (is_goal: True), a visualization of the tree will be printed to show the path traversal. Implementation of this can be seen in tree_view.py.
    - For best results, expand the cmd/terminal window to see the correct structure.
    - Example of BFS that traversed through 13 nodes:

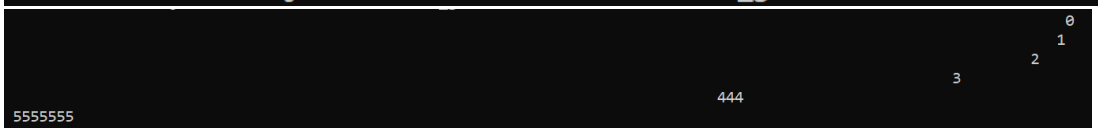
```
node: <node.Node object at 0x00000293DE164BE0>  is_goal: False  node level: 0
node: <node.Node object at 0x00000293DE172C50>  is_goal: False  node level: 1
node: <node.Node object at 0x00000293DE172C18>  is_goal: False  node level: 1
node: <node.Node object at 0x00000293DE172C88>  is_goal: False  node level: 1
node: <node.Node object at 0x00000293DE172D30>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172CF8>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172D68>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172DA0>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172CC0>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172E10>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172E48>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172DD8>  is_goal: False  node level: 2
node: <node.Node object at 0x00000293DE172EB8>  is_goal: True   node level: 2
```

    -

```
    0
  111
222222222
```

    - Example of DFS that traversed 14 nodes:

```
node: <node.Node object at 0x0000017B89344BE0>  is_goal: False  node level: 0
node: <node.Node object at 0x0000017B89352C18>  is_goal: False  node level: 1
node: <node.Node object at 0x0000017B89352CF8>  is_goal: False  node level: 2
node: <node.Node object at 0x0000017B89352DD8>  is_goal: False  node level: 3
node: <node.Node object at 0x0000017B89352EB8>  is_goal: False  node level: 4
node: <node.Node object at 0x0000017B89352F98>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B89352F60>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B89352FD0>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B89352E80>  is_goal: False  node level: 4
node: <node.Node object at 0x0000017B89352F28>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B893610B8>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B893610F0>  is_goal: False  node level: 5
node: <node.Node object at 0x0000017B89352EF0>  is_goal: False  node level: 4
node: <node.Node object at 0x0000017B89361128>  is_goal: True   node level: 5
```

- 

    - * Visualization of the DFS is a bit sparse due to the length of the 5th level, which is 243 nodes

Notes on the Implementation:

1. For this homework, we initially decided that the basis of a reached goal state is an integer generated randomly (See node class). If the basis is equal to 8, it will return True, which will cause the program to terminate. Otherwise, it will return False, and continue traversing.
2. In this implementation, each node has a 5% chance to be a goal state (1/20).
3. The children of a node are implemented as a list. This helps in making the branching factor dynamic.
    a. children[0] is implemented as the leftmost side of the list
    b. children[length-1] is implemented as the rightmost side of the list
4. DFS and BFS functions have the same implementation. The only differences are the following:
    a. Data structure that was used for each traversal type (Queue and Stack for BFS and DFS, respectively)
    b. The placement of the level tracker (Different algorithm for BFS and DFS)