Question 1. [12 MARKS]

In this question you will draw trees. We don't want you to draw a memory model; just circles with values inside them and lines connecting them like we've been doing in lecture and like you see in parts (b) and (d).

Part (a) [3 MARKS]

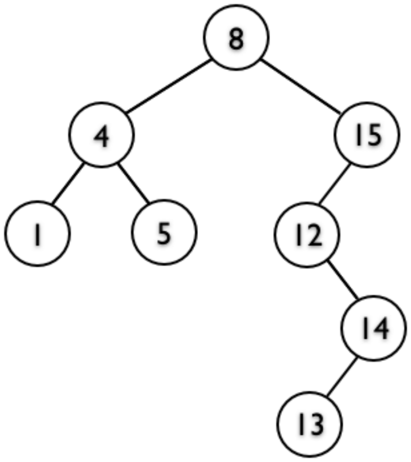
This subquestion is about **binary search trees**. Draw the tree that results from inserting the following values in order:

```
2 6 4 3 5 1 9
```

```
    2
  1     6
      4   9
     3 5
```

Part (b) [3 MARKS]

This subquestion is about **binary search trees**. In the blank space on the right, draw the tree that results from deleting the root value from the following tree, where we replace a deleted value with the smallest value from the right subtree:



```
    12            12
  4     15 OR    4     15
 1 5   14       1 5  13
      13             14
```

Part (c) [3 marks]

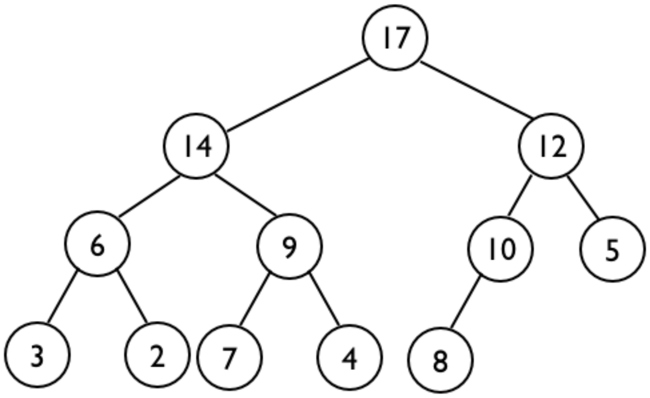
This subquestion is about **min heaps**. Draw the tree that results from inserting the following values in order into a min heap.

```
2 6 4 3 5 1 9
```

```
    1
  3   2
 6 5 4 9
```

Part (d) [3 marks]

This subquestion is about **max heaps**. In the blank space on the right, draw the tree that results from deleting the root value from this tree:



```
        14
    9       12
  6   8   10  5
 3 2 7 4
```

Question 2. [5 marks]

A file storing the pre-order, post-order and in-order traversals of a **binary tree** has been corrupted and most values have been lost. Below, these "lost" values are replaced with the letter `x`. Draw the binary tree that produced the file. (If you think more than one tree matches these traversals, you can draw any of them.)

```
  pre-order:  x,  9,  2, 12,  x,  5,  x
 post-order:  x,  x,  7,  x,  1,  x,  6
   in-order:  2,  x,  9,  7,  6,  1,  x
```

```
              6
     9              5
2        7       1
  12
```

Question 3. [8 marks]

We have worked extensively with binary trees. This question is about trees with an arbitrary branching factor.

Part (a) [3 marks]

Write a `Node` class that allows any number of children. Write only the class header and an `__init__` method that sets up any instance variables.

```
class Node(object):
    def __init__(self, v):
        self.key = v
        self.children = []
```

Part (b) [5 marks]

Write a function that prints the values in a tree with an arbitrary branching factor using a preorder traversal. The tree is made from instances of your `Node` class from the previous subquestion. You can directly access any instance variables that you need.

```
def traverse(r):
    if r:
        print r.key
        for sub in r.children:
            traverse(sub)
```

Question 4. [8 marks]

Two trees are *mirrors* of each other if they have the same contents but the left and right children are swapped throughout. For example, these two trees are mirrors of each other:

```
          1                       1
        /   \                   /   \
      2       3               3       2
     /      /   \           /   \      \
    4      5     6         6     5      4
```

Complete the following function[1].

```
    def are_mirrors(root1, root2):
        '''(Node, Node) -> bool
        Return whether the trees rooted at root1 and root2 are mirrors of each other.'''

        return (not root1 and not root2) or (
            root1 and root2 and
            root1.key == root2.key and
            is_mirror(root1.left, root2.right) and
            is_mirror(root1.right, root2.left)
        )
```

[1] If you have extra time and want a fun challenge, try doing it using only a single return statement and no ifs or loops.