CSC 226 - Fall 2014
Pseudocode for a Union-Find Data Structure

The pseudocode listings below give the three core operations for a union-find structure with union-by-rank and path compression. The listings on the left use an object-based linked structure and the listings on the right use a flat array representation. For clarity, the arrays in the right-hand listings are treated as global variables. An actual implementation should avoid this and instead have the `parent` and `rank` arrays be accepted as arguments to each function. The array-based version requires that the maximum number of elements n be fixed in advance (since the arrays have a fixed size).

Note: The term 'rank' is not defined consistently between sources. Depending on the interpretation, it may refer to the number of items in a set or the height of the corresponding union-find tree. Both interpretations give the same running times in a union-find structure when used consistently. Formally, the term 'rank' refers to the size of a set. Pseudocode for both versions are given here.

The version below uses 'rank' to refer to set size.

```
1:
2:
3:
4:  procedure MakeSet(k)
5:      node ← new Node
6:      node.value ← k
7:      node.rank ← 0
8:      node.parent ← node
9:      return node
10: end procedure
11:
12: procedure Find(X)
13:     if X.parent = X then
14:         return X
15:     else
16:         X.parent ← Find(X.parent)
17:         return X.parent
18:     end if
19: end procedure
20:
21: procedure Union(X, Y)
22:     X ← Find(X)
23:     Y ← Find(Y)
24:     if X.rank > Y.rank then
25:         Y.parent ← X
26:         X.rank ← X.rank + Y.rank
27:     else
28:         X.parent ← Y
29:         Y.rank ← X.rank + Y.rank
30:     end if
31: end procedure
```

```
parent ← Array of n integers.
rank ← Array of n integers.

procedure MakeSet(k)
    parent[k] ← k
    rank[k] ← 0
end procedure

procedure Find(x)
    if parent[x] = x then
        return x
    else
        parent[x] ← Find(parent[x])
        return parent[x]
    end if
end procedure

procedure Union(x, y)
    x ← Find(x)
    y ← Find(y)
    if rank[x] > rank[y] then
        parent[y] ← x
        rank[x] ← rank[x] + rank[y]
    else
        parent[x] ← y
        rank[y] ← rank[x] + rank[y]
    end if
end procedure
```

The version below uses 'rank' to refer to the height of the union-find tree.

```
1:
2:
3:
4:  procedure MakeSet(k)
5:      node ← new Node
6:      node.value ← k
7:      node.rank ← 0
8:      node.parent ← node
9:      return node
10: end procedure
11:
12: procedure Find(X)
13:     if X.parent = X then
14:         return X
15:     else
16:         X.parent ← Find(X.parent)
17:         return X.parent
18:     end if
19: end procedure
20:
21: procedure Union(X, Y)
22:     X ← Find(X)
23:     Y ← Find(Y)
24:     if X.rank > Y.rank then
25:         Y.parent ← X
26:     else
27:         X.parent ← Y
28:         Y.rank ← max(Y.rank, X.rank + 1)
29:     end if
30: end procedure
```

```
parent ← Array of n integers.
rank ← Array of n integers.

procedure MakeSet(k)
    parent[k] ← k
    rank[k] ← 0
end procedure

procedure Find(x)
    if parent[x] = x then
        return x
    else
        parent[x] ← Find(parent[x])
        return parent[x]
    end if
end procedure

procedure Union(x, y)
    x ← Find(x)
    y ← Find(y)
    if rank[x] > rank[y] then
        parent[y] ← x
    else
        parent[x] ← y
        rank[y] ← max(rank[y], rank[x] + 1)
    end if
end procedure
```