Eli Sobylak
Chapter 8

16-Apr-16
DS

8.1:

b) Implicit Representation

16	2	4	4	-11	4	4	2	9	2	4	4	4	4	16	4	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

c) Implicit Rep.

-1	2	7	4	5	6	10	9	9	13	11	12	13	15	15	-1	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

8.2:

c)

-1	2	7	15	15	15	15	9	9	13	15	15	15	15	15	-1	15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

```
#include <iostream>
#include <vector>

using namespace std;

class DisjSets
{
  public:
    explicit DisjSets( int numElements );

    int find( int x ) const;
    int find( int x );
    void unionSets( int root1, int root2 );

  private:
    vector<int> s;
};

/**
 * Construct the disjoint sets object.
 * numElements is the initial number of disjoint sets.
 */
DisjSets::DisjSets( int numElements ) : s( numElements )
{
    for( int i = 0; i < s.size( ); i++ )
        s[ i ] = -1;
}

/**
 * Perform a find.
 * Error checks omitted again for simplicity.
 * Return the set containing x.
 */
int DisjSets::find( int x ) const
{
    if( s[ x ] < 0 )
        return x;
    else
        return find( s[ x ] );
}

/**
 * Perform a find with path compression.
 * Error checks omitted again for simplicity.
 * Return the set containing x.
 */
```

```
int DisjSets::find( int x )
{
   if( s[ x ] < 0 )
      return x;
   else
      return s[ x ] = find( s[ x ] );
}

/**
 * Union two disjoint sets.
 * For simplicity, we assume root1 and root2 are distinct
 * and represent set names.
 * root1 is the root of set 1.
 * root2 is the root of set 2.
 */
void DisjSets::unionSets( int root1, int root2 )
{
   if( s[ root2 ] < s[ root1 ] )  // root2 is deeper
      s[ root1 ] = root2;       // Make root2 new root
   else
   {
      if( s[ root1 ] == s[ root2 ] )
         s[ root1 ]--;          // Update height if same
      s[ root2 ] = root1;       // Make root1 new root
   }
}


void printElementSets(const DisjSets & s)
{
          for (int i = 0; i < s.NumElements(); ++i)
                    cout << s.FindSet(i) << "  ";
          cout << endl;
}

int main()
{
          DisjSets(10);
          printElementSets(s);
          s.Union(s.FindSet(5),s.FindSet(3));
          printElementSets(s);
          s.Union(s.FindSet(1),s.FindSet(3));
          printElementSets(s);
          s.Union(s.FindSet(6),s.FindSet(7));
          printElementSets(s);
          s.Union(s.FindSet(8),s.FindSet(9));
```

```
        printElementSets(s);
        s.Union(s.FindSet(6),s.FindSet(9));
        printElementSets(s);
        s.AddElements(3);
        printElementSets(s);
        s.Union(s.FindSet(11),s.FindSet(12));
        printElementSets(s);
        s.Union(s.FindSet(9),s.FindSet(10));
        printElementSets(s);
        s.Union(s.FindSet(7),s.FindSet(11));
        printElementSets(s);

        system("pause");
        return 0;
}
```