CS 2420 Program 6 – 20 points
Fall 2015 Union Find

Part 1: Union Find

Write the code to perform union find (using path compression) on a set of 30 elements. Use a smart union (either by size or by height, your choice.) Create a series of carefully constructed tests so that you can verify union/find is working properly and that path compression works. Print out the contents of your array.

Everyone's tests will be different. You will be graded on how well your tests demonstrate properly functioning. This ability to test your code (and dig into the details) is exactly the skill referred to by "Don't debug in the dark".

For ease in programming (and for an easy transition to part 2), instead of using the same variable to be EITHER parent or negative size, consider using an array of pairs as shown below.

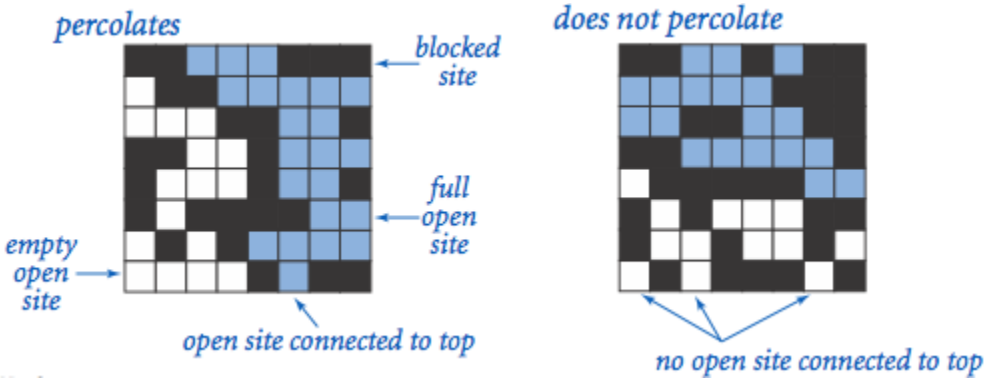
	ParentID	Size
0	0	1
1	1	1
2	4	
3	4	
4	4	7
…		
19	19	6

Part 2: Percolation

Percolation. Given a porous landscape with water on the surface under what conditions will the water be able to drain through to the bottom? Scientists have defined an abstract process known as *percolation* to model such situations.

The model. We model a percolation system using an *N*-by-*N* grid of *sites*. Each site is either *open* or *blocked*. A *full* site is an open site that would fill with water if water were applied to the top of a grid. In other words, a full site can be connected to an open site in the top row via a chain of neighboring (left, right, up, down) open sites. We say the system *percolates* if there is a full site in the bottom row. In

other words, a system percolates if we fill all open sites connected to the top row and that process fills some open site on the bottom row



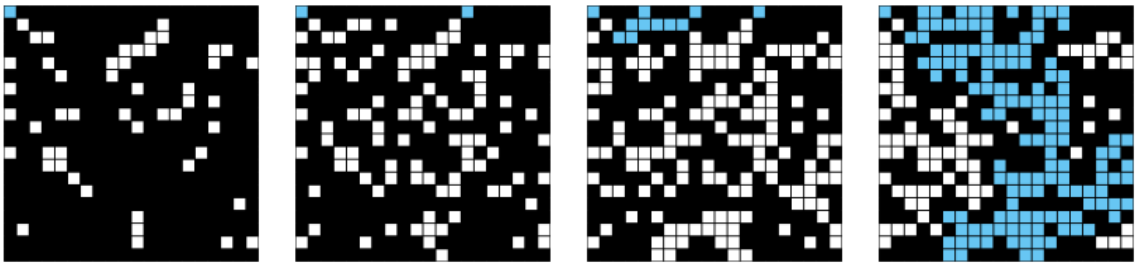
The problem. In a famous scientific problem, researchers are interested in the following question: if sites are independently set to be open with probability p (and therefore blocked with probability $1 - p$), what is the probability that the system percolates?

Monte Carlo simulation. To estimate the percolation threshold, consider the following computational experiment:

- Initialize all sites to be blocked.
- Repeat the following until the system percolates:
  - Choose a site (row i, column j) uniformly at random among all blocked sites. [If you select an already open site, try again.]
  - Open the site (row i, column j).
- The fraction of sites that are opened when the system percolates provides an estimate of the percolation threshold.

By repeating this computation experiment T times and averaging the results, we obtain a more accurate estimate of the percolation threshold.

For example, if sites are opened in a 20-by-20 grid according to the snapshots below, then our estimate of the percolation threshold is 204/400 = 0.51 because the system percolates when the 204th site is opened.



50 open sites *100 open sites* *150 open sites* *204 open sites*

In a readme.txt file, answer the following questions regarding the analysis of running time and memory usage.

Output:

Using a 20-by-20 grid, (1) show the grid after every 50 sites have been opened and (2) show the grid when it finally percolates. Each site should be printed using the following notation:

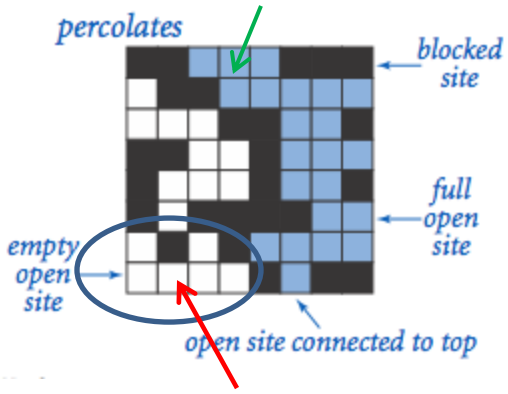
a. a blank space for open
b. X for blocked
c. The edges of the grid are marked with '_' or '|'

Hints:

1. You won't actually denote sites which are connected to the top (or bottom) differently – as union/find can't easily tell you which elements are members of the same group.
2. Use the smart union find (with path compression) to keep track of the set of sites that are connected.

   Given the groups, you will need to plan to decide how you can tell if a group percolates. You can do this any way you like, as long as it is efficient. Our goal with union/find is to have an almost constant time operation. Thus, anything you add has to maintain this goal.

   One idea is to have each root keep needed information. If the root of a group always stores the lowest and highest rows in the collection, it will be easy to tell when it percolates.



So, for the case above, the array might contain more information. 57 is the number of the site pointed to by the red arrow. It is the root of a group of 6 elements which are between row 6 and 7. The green arrow points to site 4.

	ParentID	Size	lowestRow	highestRow
0	0	1	0	0
1	1	1	0	0
2	4			
3	4			
4	4	22	0	7
…				
57	57	6	6	7

Another way of accomplishing the goal is to have a dummy row at the top and a dummy row at the bottom of the grid. The top dummy row will be unioned into the same group. The dummy bottom row will be unioned into a different group. After opening up nodes in the grid, the whole grid percolates if anything in the top dummy row is in the same group as anything in the bottom dummy row.

3. Since union find works on integers (instead of pairs of integers), you will want a way of converting between (x,y) and an integer, i. One way to do this is as shown below:
   a. Given x and y i = N*x+y
   b. Given i, x = i/N and y = i%N

This assignment was developed by Bob Sedgewick and Kevin Wayne.