The first page of your homework submission must be a cover sheet answering the following questions. Do not leave it until the last minute; it's fine to fill out the cover sheet before you have completely finished the assignment. Assignments submitted without a cover sheet, or with a cover sheet obviously dashed off without much thought at the last minute, will not be graded.

- How many hours would you estimate that you spent on this assignment?

- Explain (in one or two sentences) one thing you learned through doing this assignment.

- What is one thing you think you need to review or study more? What do you plan to do about it?

Question 1. Prove: for all $n \geq 1$, if G is a connected graph with n vertices and $n-1$ edges, then G has no cycles. (This is the third part of the proof from class, characterizing trees as having any two out of three properties.)

1

Question 2. Let $G = (V, E)$ be an undirected graph with n vertices, with no self-loops (that is, no edges of the form (v, v) from a vertex to itself). Show that if every vertex has degree at least $n/2$, the graph is connected. If it makes your proof easier, you may assume that n is even.

2

Question 3. Consider the family of undirected graphs $\mathcal{H}_k$ defined as follows. $\mathcal{H}_k$ has 2^k vertices labelled with the integers 0 through $2^k - 1$. Vertices u and v are connected by an edge if and only if the binary representations of u and v differ in exactly one bit position. For example, in $\mathcal{H}_4$, the vertices 5 and 13 are connected by an edge since $5 = 0101_2$ and $13 = 1101_2$ differ in the first bit position, but the rest of the bits are the same.

Consider doing a BFS in $\mathcal{H}_{10}$ starting at node 0. How many vertices are in L_6, that is, the sixth layer generated by the BFS? Give your answer together with either a proof, or the program you used to calculate the answer. Either approach will receive full credit. (*Hint* if you choose to write a program: to flip the `j`th bit of an integer `n`, you can use `n ^ (1 << j)`, that is, the bitwise XOR of `n` with the result of shifting 1 left j times, that is, 2^j. These operators are valid syntax in many languages such as Java, Python, and C.)

Question 4.

- Visit `http://open.kattis.com/` and make an account if you don't already have one.
- Solve `http://open.kattis.com/problems/torn2pieces` using any programming language you want. I have provided starter code in Python and Java on the course website.
- Submit a printout of your code, along with a link to your user profile on Kattis.

I am well aware that there are many solutions to this problem that can be found on the Internet. Please **do not look at them**. You will learn far less if you look at someone else's solution, not to mention that it is an academic integrity violation—*even* if you only intend to "get a few ideas" and then write your own solution.

A few hints:

- If you want to use a BFS, you don't need to keep track of the layers as in the pseudocode given in class. Instead, you can just keep a queue of vertices to be processed. That is, instead of putting a node in L_{i+1}, just put it in the queue. Keep taking vertices out of the queue and processing them until the queue is empty. Using a queue ensures that we finish processing each layer completely before moving on to any vertices in the next layer.

- Note that there may be stations which are not connected to *any* other stations (although this would be quite strange in real life)!
- Be sure to process each connection as a *two-way* (undirected) connection. Some stations may not be listed separately if all their connections have already been listed under other stations. For example, the input

```
2
A C
B C
A B
```

  should yield the output `A C B`. Station `C` is not listed on its own line, but it shows up as being connected to both `A` and `B`.
- You may or may not find it helpful to make a separate `Graph` class. When submitting code to Kattis you cannot submit multiple files, but you can put multiple classes in the same file (even in Java!—just make sure that only one of the classes is marked `public`).

Extra credit: Tree diameter

Question 5 (Optional extra credit)**.** As in the text, we say that the *distance* between two nodes u and v in a graph $G = (V, E)$ is the minimum number of edges in a path joining them; we'll denote this by $dist(u, v)$. We say that the *diameter* of G is the maximum distance between any pair of nodes, that is,

$$diam(G) = \max_{u,v \in V} dist(u, v).$$

Give a $\Theta(|V| + |E|)$-time algorithm to find the diameter of an (undirected) **tree** $T = (V, E)$. Prove that your algorithm is correct for trees, and give a counterexample showing that your algorithm does not correctly find the diameter of all graphs.

5

Be careful that you understand what this definition says—the diameter is *not* the length of the longest possible path! It is the maximum length of a *shortest* path. These sorts of maximum-of-minimum definitions can be tricky to think about, but they come up all the time—as another example, for a given problem we typically want to come up with the algorithm with the *best worst*-case behavior.