In this assignment, you are asked to implement the RANDMINCUT algorithm for finding minimum cuts in undirected graphs from the Feb 4 lecture [also, MU Section 1.4]:

1. Repeat $|V| - 2$ times: contract a random edge
2. Output the edges connecting the remaining two vertices.

Program Input/Output. The main subroutine in your program should be an implementation of RANDMINCUT. The program should accept as input a text file specifying an undirected graph, in the following format: the first line specifies the number of vertices, and each additional line specifies an edge. For instance, the following text corresponds to the graph $V = \{1, 2, 3, 4, 5, 6\}$, $E = \{(1, 2), (2, 3), (1, 3), (3, 4), (4, 5), (5, 6), (4, 6)\}$ (a future version of this assignment may require that you support additional graph formats):

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

The output of your program should be a cut in the input graph obtained by running a single iteration of RANDMINCUT (which may not be a minimum cut). The first line should specify the edges in the cut-set, followed by the disconnected components in the graph upon removing these edges. For the above example, the output may look like:

```
(5,6) (4,6)
1 2 3 4 5 | 6
```

Command-Line Arguments. The first command-line argument passed to your program will be the name of a text file specifying the graph. An optional second argument specifies the number of iterations of RANDMINCUT to run. Moreover, if the final (optional) argument is `-v`, then the program runs in verbose mode, specifying the edges that are being contracted and the self-loops that are removed upon contradiction, e.g.:

```
% java RandMinCut test1.dat -v
contracting (2,3)
removing self-loop (2,3)
contracting (5,6)
removing self-loop (5,6)
```

```
contracting (4,5)
removing self-loop (4,5)
removing self-loop (4,6)
contracting (1,2)
removing self-loop (1,2)
removing self-loop (1,3)
(3,4)
1 2 3 | 4 5 6

% java RandMinCut test2.dat -v
contracting (1,2)
removing self-loop (1,2)
contracting (2,3)
removing self-loop (2,3)
empty
1 2 3 | 4 | 5

% java RandMinCut test3.dat
(81,82) (50,140) (50,141) (2,161)
[...]
```

Implementation Tips. You may find the disjoint-set data structure and the `union-find` algorithm handy for implementing edge contractions. An overview and an implementation of the `union-find` algorithm are linked on the course webpage.