Assignment 1: Frequently Asked Questions

Question 1. **(Set ordering)** How should vertices and edges be ordered? The assignment says "Your output has to perfectly match what is expected."

Answer. Since the vertices V and edges E are sets, there is no specification on ordering. You may output the entries in any order. The graph is not directed, so edges themselves, i.e., pairs of coordinates, can be output in arbitrary order as well.

Question 2. **(Intersections)** Should we calculate the intersection points in floating point format or integer?

Answer. The intersections are not necessarily integers. You can, for example, assume that coordinate values within, say, 0.0001 of one another are the same. We will not test your code with weird intersections that are very close to one another.

Question 3. **(White-space)** For the following examples, can you tell me if the code should accept it or reject it?

```
add "weber" 1,2,3,4,5,6 - no brackets
add "weber" (1,2 (3,4) (5,6) - missing bracket
add "weber" (1,2)(3,4) (5,6) - no space between the brackets
add"weber" (1,2) (3,4) (5,6) - missing space between command
add "weber"(1,2) (3,4) (5,6) - missing space after street name
```

Answer. The first and second examples are erroneous and should be rejected. The third example is valid. The fourth and fifth examples are erroneous and should be rejected because there is missing space that does not separate the street from the rest of the command line. This is typical with software – we often ignore the exact number of white-space characters (or all white-space altogether). The same is true for the output graph that your program must generate.

Question 4. **(Float format)** How should float values be printed. Is the following a good output?

```
g
V = {
  1: (2.000000,2.000000)
  2: (4.000000,4.000000)
}
E = {
  <1,2>
}
```

Answer. No. You should print all coordinate values rounded to two decimal places. You can do so using `format` function when converting a `float` to a `string`. For example,

```
>>> "{0:.2f}".format(2.0)
'2.00'
>>> "{0:.2f}".format(2.324)
'2.32'
```

Note that it is not necessary to always have two decimal digits. For example, an integer coordinate 2 can be printed as either one of `2`, `2.0`, or `2.00`.

Question 5. Do vertex ids have to be numeric. Should they always be printed in ascending (smallest-to-largest) order?

Answer. No, vertex ids can be arbitrary strings subject to the specification in the assignment.

Question 6. Can a street intersect itself?

Answer. We assume that streets do not intersect themselves. Note that street segments of different streets may overlap. In the case of overlapping street segments, the end points of the overlapping region are vertices.

Question 7. Are street names case sensitive? Do spaces matter in street names?

Answer. Street names are case insensitive. For example, "King street" and "KING street" mean the same street. The street name is composed of all the characters between quotes. If the characters are different, the street names are different. This includes whitespace characters as well. For example, "King" is a different from " King".

Question 8. Are command names lower case?

Answer. Command names are lower case as in all examples in the handout.

Question 9. What characters are allowed in street names?

Answer. Alphabetical and space characters only. No numbers, commas, special characters, unicode characters, etc.

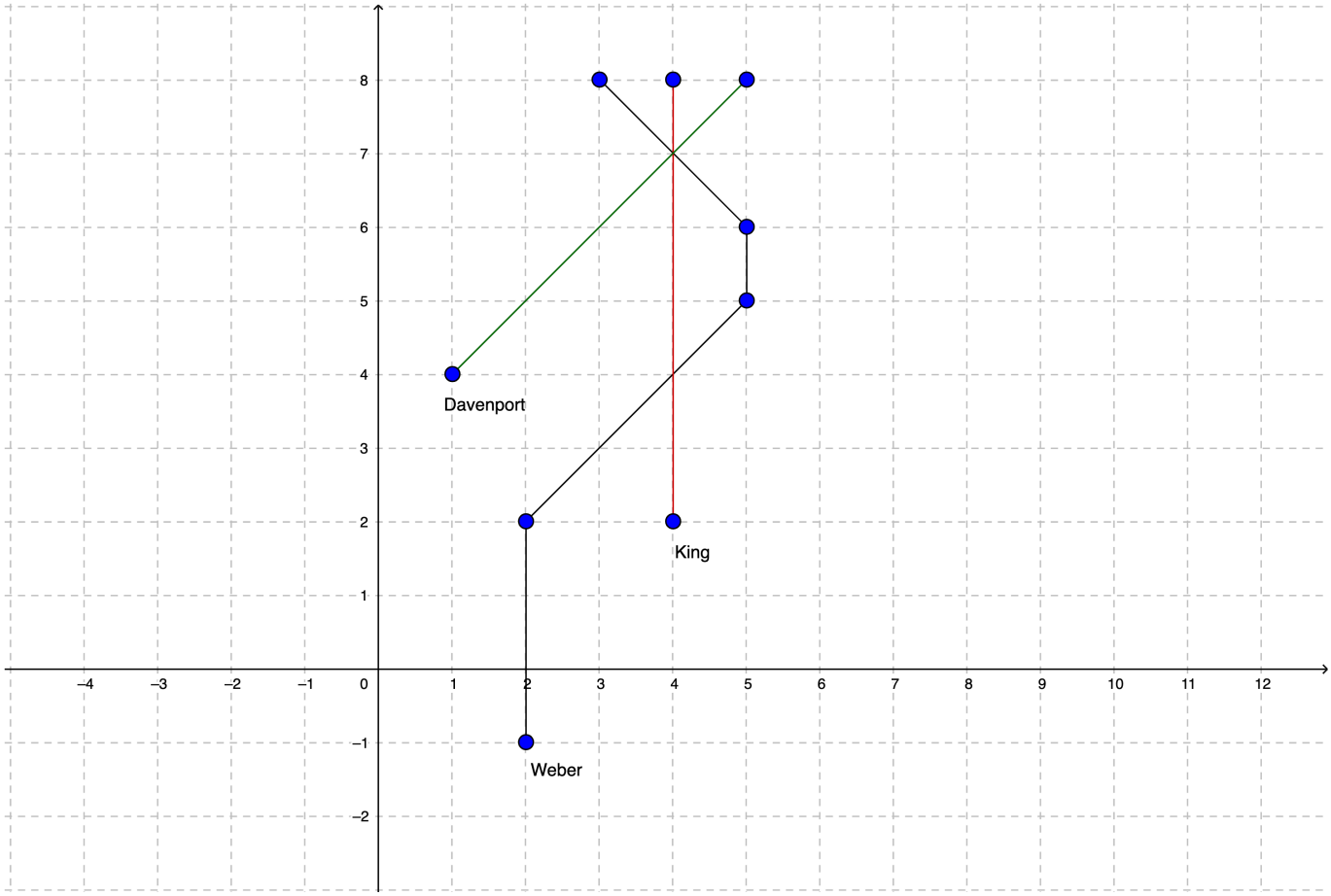
Question 10. How do we find the intersection between two line segments?

Answer. There is an example in Python in the repo `https://git.uwaterloo.ca/ece650-1209/py`. While this produces an intersection point, be careful and consider if the intersection point makes sense or is valid.

Question 11. What does the format of the edges mean?

Answer. The edges are represented as a connection or edge between two verticies. In the following example the edge is a connection between vertex 1 and 2.

```
g
V = {
  1: (2.000000,2.000000)
  2: (4.000000,4.000000)
}
E = {
  <1,2>
}
```


Davenport
King
Weber