The order in which mesh components are listed determines which component is being referred to by a face definition line:
- The first vertex position is “index 1” NOT “index 0”

# denotes a comment line

**v** 1.00 1.00 1.00 denotes a vertex located at <1,1,1>

**vn** 1.00 0.00 0.00 denotes a vertex normal with a direction of <1,0,0>

**vt** 1.00 0.00 denotes a vertex texture coordinate located at <1,0>

Faces are ALWAYS defined by a ring of counter-clockwise vertices:
- Convenient since your half-edge meshes are also counter-clockwise
The bottom two lines each define a face
The faces are composed of vertex positions defined by the indices on their definition lines

```
v 1 1 0
v 0 1 0
v 1 0 0
v 0 0 0
f 1 2 4
f 1 4 3
```
OBJ Format: Faces

v 1 1 0
v 0 1 0
v 1 0 0
v 0 0 0
f 1 2 4 3

OBJ files can define faces to have any number of sides, so the face definition above is also valid.
OBJ Format: Faces

v  1  1  0
v  0  1  0
v  1  0  0
v  0  0  0
vt 1  1
vt 0  1
vt 1  0
vt 0  0
f 1/1 2/2 4/4 3/3

This face is of the format position/texture coordinate
This face is of the format **position*/texture coordinate*/normal**
This face is of the format `position//normal`
Qt has several classes and functions that are useful for file reading

- **QFileDialog::getOpenFileName** lets you open a file browser to choose an OBJ file to read. It returns a QString.
- A **QFile** can be constructed, given a filename.
  - Among other functions, it has the `open()` method which lets you begin parsing a file.
- A **QTextStream** can be used to read lines of a QFile one at a time.
  - They can also be used to store information in variables using the `>>` operator. This uses whitespace to distinguish tokens.

```cpp
int x, y, z;
QTextStream stream = QTextStream("1 2 3");
stream >> x >> y >> z;
```
You can also use C++ functions
  - `sscanf` is useful for reading lines with a fixed number of elements
    ```
    int x; char c[2]; float y;
    char* line = "1 Hi 3.14";
    sscanf(line, "%i %s %f", &x, c, &y);
    ```
  - Note that `sscanf` must take fixed-size variables as inputs, hence the initialization of `c` as an array of two chars.
Tips for converting to half-edge mesh

- Make sure that you only make one vertex per position defined by the OBJ file.
- Create all the faces of your mesh, then worry about creating sym pointers.
  - The faces of the OBJ file are not guaranteed to be defined in any sort of order, so there’s no sense in trying to connect up faces as you make them.
- It may be helpful to store a map of vertex -> list of half-edges that point to that vertex.
  - This makes creating half-edge sym pointers much easier.