CS130B Project3 Report
Shunji Zhan

In this project, I used three struct and one class to represent the whole 3d graph:

- Point struct, which represent a point
- Plane struct, which represent either upper plane of lower plane
- ToroidalDot struct, which represent a dot in the Toroidal Graph.
- Triangulation class, which is the whole thing to read in data, calculate and print the triangles.

Triangles are found in the following steps:
1, read and parse the points, build the data structures to represent the data.

2, build the Toroidal Graph. Each dot in this graph can only be accessed from its top dot or its left dot, the I used the principle of optimality:

```
thisDot.minDistance = min {
    leftDot.minDistance + distance(leftDot, thisDot),
    topDot.minDistance + distance(topDot, thisDot),
}
```

And the recurrence relation:

```
for all dots in the Toroidal Graph {
  thisDot.minDistance = min {
      leftDot.minDistance + distance(leftDot, thisDot),
      topDot.minDistance + distance(topDot, thisDot),
  }
}
```

I calculate the dots in this order: (0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0) ··· in order the made sure when calculating minDistance the topDot and leftDot's minDistance is already calculated.

The table of partial solution is just the Toroidal graph, which is something like this:

	Lower 0	Lower 1	···	Lower n
Upper 0	minDistance = 0 distanceToRight = ··· distanceToDown = ···	minDistance = ··· distanceToRight = ··· distanceToDown = ···	···	···
Upper 1	minDistance = ··· distanceToRight = ··· distanceToDown = ···	···	···	···
···	···	···	···	···
Upper m	···	···	···	···

3, after building the Toroidal Graph, I backtracked from the last dot in Toroidal Graph, which is Toroidal[m][n], and see which dot lead to this dot, until go back to Toroidal [0][0], the origin. In each backtracking step, I printed out the corresponding triangle.