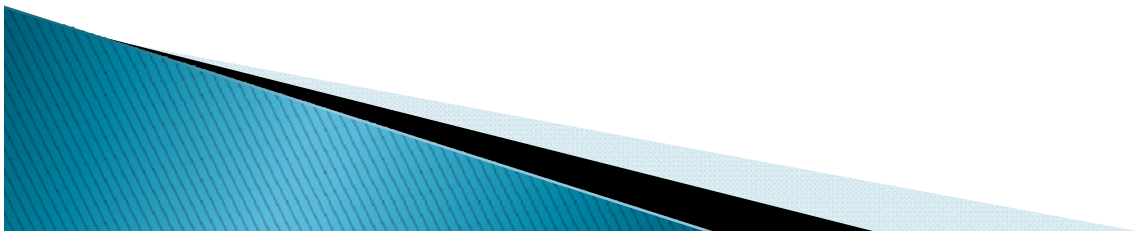
Project Phase III

Sample Input

`0000:DY 1`	*Set a one second delay for each operation shown*
`0001:IP -100 0`	*Add a point (-100,0) (index = 1)*
`0002:IP 100 0`	*Add a point (100,0) (index = 2)*
`0003:IP 0 10`	*Add a point (0,100) (index = 3)*
`0004:IP 0 -100`	*Add a point (0,-100) (index = 4)*
`0005:CD`	*Compute the Delaunay Triangulation of the input points*
`0006:IP 50 0`	*Add more points*
`0007:IP 30 2`	
`0008:CD`	*Compute the Delaunay Triangulation of the previous triangulation and the new additional points together*



Project Phase III

- Implement the edge flipping algorithm
  - But you are **not** required to follow the one in the lecture exactly
- With Phase II, read in a file "input.txt"
  - And animate it, show each step of instruction, including each flip
  - The "artistic" aspects of the animation is also graded
- When the animation is finished
  - Insert a point ("IP") by a **right click**
  - Compute and display the Delaunay Triangulation by "c" or "C"
- Finally, press "w" to output a file with all the points (only, without any triangle) with "IP" and a final CD command.
- You can assume no duplicated points and no more than four points are co-circular
  - But if you can handle these, there will be bonus
- Still, expect the unexpected

Time Stamps

- Please put time stamps before and after each time you compute the Delaunay triangulation and output the timing (in milliseconds) to "cerr"

```
SYSTEMTIME st;

GetLocalTime(&st);
int start = (((st.wHour*60+st.wMinute)*60)+st.wSecond)*1000+st.wMilliseconds;
cerr << "Start: " << start << endl;


computeDelaunayTriangulation();


GetLocalTime(&st);
int end = (((st.wHour*60+st.wMinute)*60)+st.wSecond)*1000+st.wMilliseconds;
cerr << "End: " << end << endl;

cerr << "Elapsed Time(ms): " << end-start << endl;
```