CS310 Computational Geometry
Spring 2019

Assignment:	**4**	**Professor:**	**Dianna Xu**
Due Date:	**3/19/19**	**E-mail:**	`dxu@cs.brynmawr.edu`
Office:	**Park 203**	**URL:**	`http://cs.brynmawr.edu/cs310`

Assignment 4

1. Exercise 3.3

2. Exercise 3.21 (a) and 3.26 (a)

3. Exercise 3.52

4. You are given two convex polygons P^- and P^+, where P^- is enclosed within P^+. Let n be the total number of vertices between these two polygons. Present an efficient (ideally O(n)) algorithm that computes a convex polygon Q whose boundary is nested between P^- and P^+ (see Figure 1). As usual, you should state the running time and have proof of correctness.

In addition, your polygon Q should have the smallest number of sides possible. More formally, let Q^* be the polygon nested between P^- and P^+ with the smallest number of sides. Prove that, if Q^* has k sides, then the polygon that your algorithm produces has at most ck sides, for some constant c (which does not depend on n).

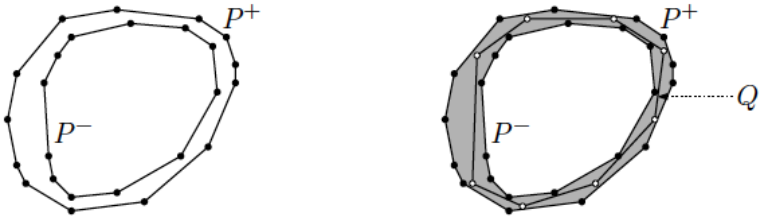

Figure 1: Polygon Q

5. In computer graphics, triangulations are used to represent surfaces. Graphics libraries like OpenGL support a special type of triangulation called a *strip*, which is defined as follows. For $n \geq 3$, given a sequence of vertices $V = \langle v_1, v_2, \ldots, v_n \rangle$, the strip of V consists of the $n-2$ triangles $\langle v_i, v_{i+1}, v_{i+2} \rangle$, for $1 \leq i \leq n-2$. See Figure 2.

Assume that you are given an n-sided, x-monotone simple polygon P as a sequence of vertices in counterclockwise order.

1. Give an algorithm to decide if P can be triangulated as a strip of the form given above.
2. Give a proof of correctness and derive the running time of your algorithm.

Note: The x-monotonicity will play a significant role in your algorithm and its proof of correctness. There is a simple linear time algorithm for this problem, but the proof of correctness is not completely straightforward. Your algorithm need not run in linear time. However, a more efficient algorithm will receive more credit.

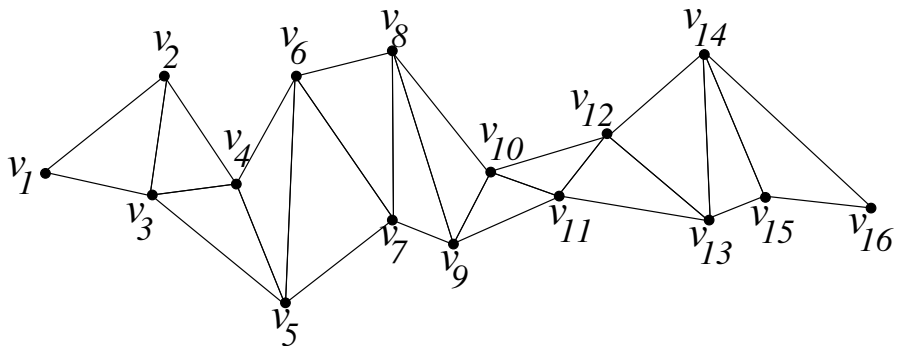

Figure 2: An x-monotone strip polygon.

6. Write a program to generate and count the number of triangulations of a convex n-gon. The count-answer should be the Catalan number C_{n-2}. Your program should list the diagonals in each triangulation, but have an option to suppress the listing and just count (so that it can be checked for large n. Don't go crazy (in terms of the size of n) however, as the Catalan numbers are exponential). If you want to be specific, you can use the regular n-gon, but all convex n-gons have the same set of and number of triangulations. However you end up doing this is fine with me. As long as you list the diagonals in each triangulation, and list each triangulation exactly once, you have satisfied the task. As an example, your output for $n = 5$ could be (if you index the vertices $0, 1, 2, 3, 4$, see Figure 3.12 (a) in your book):

```
(0,2), (0,3)
(1,3), (1,4)
(2,4), (2,0)
(3,0), (3,1)
(4,1), (4,2)
```

Either way, just a list of the diagonal endpoint indices (not coordinates!) in some order (any order). No graphics is needed (but you probably want to generate graphics to make debugging easier). The challenge is to come up with a way to count for the biggest n you can manage. Because of the exponential growth and the requirement to list diagonals, you need to consider your algorithm and data structure choices. The primary one is are you storing the diagonals or not? If you store, how?