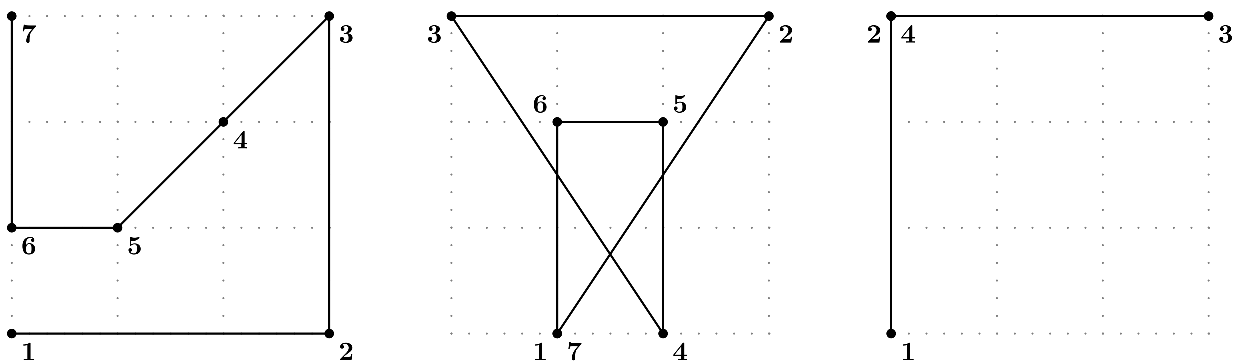
CS 130, Homework 7

Name: ______________________________ ID: ____________

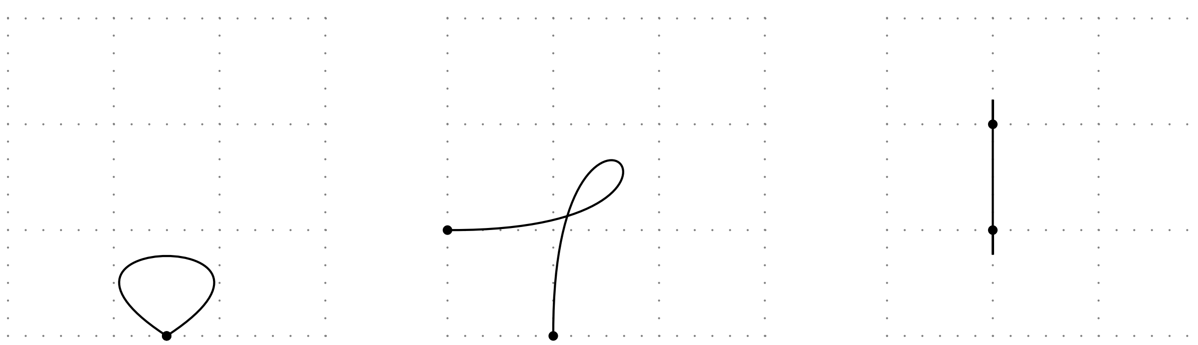
Problem 1

In each of the three examples below, control points are shown for one or two cubic Bezier curves. (Two curves share the middle point, so there are 7 points rather than 8.) Sketch out approximately what these curves will look like. Be sure to draw your sketch over a grid with control points labeled.



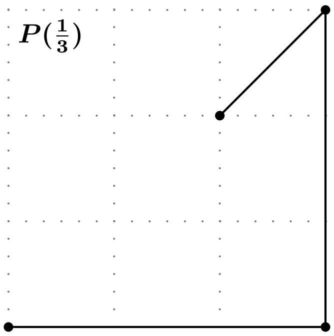
Problem 2

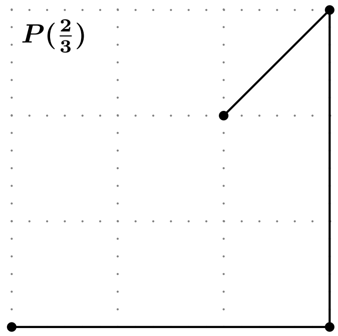
For each cubic Bezier curve below, estimate the locations of the control points. The dots show the endpoints of the Bezier.



Problem 3

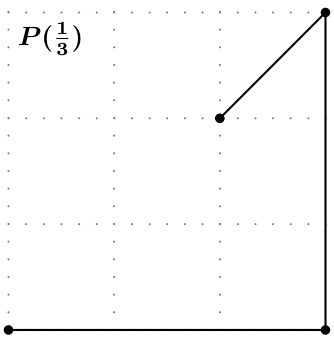
Geometrically construct the location of $P(t)$ for the Bezier curve $P(t)$ below. Note that $P(0)$ is at the bottom left in each case.

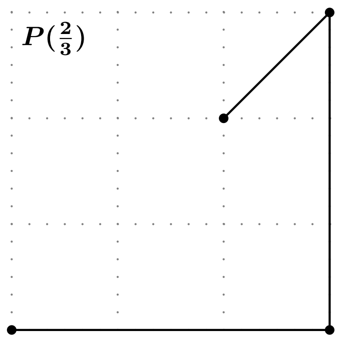




Problem 4

Subdivide the Bezier at the points chosen. Label the control points of the subdivided curves (1-7), with 1 at $P(0)$ at the bottom left.





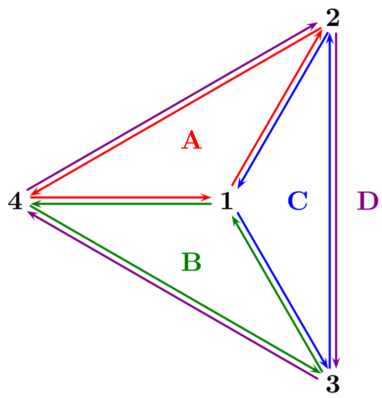
Problem 5

The half-edge structure has lots of invariants (things that should always be true). An implementation should test these invariants when it creates or modifies the structure to catch mistakes early. Assume `e` is a coedge, `v` is a vertex, and `f` is a face. Fill in the missing attribute names to make each assertion true (there may be more than one correct answer). You may find these useful for the problems that follow.

(a) `assert(e==e->pair->________);`
(b) `assert(e==e->next->________);`
(c) `assert(e==e->prev->________);`
(d) `assert(e->face==e->next->________);`
(e) `assert(e->head==e->pair->________);`
(f) `assert(e->head==e->next->________);`
(g) `assert(v==v->edge->________);`
(h) `assert(f==f->edge->________);`

Problem 6

Below is a half-edge structure for a tetrahedron, viewed from above. (D is the bottom face.) Fill in the pointers in the tables below. Follow the coedge labeling convention from class (e.g., the face of AB is A) and the convention of choosing outgoing edges for vertices.



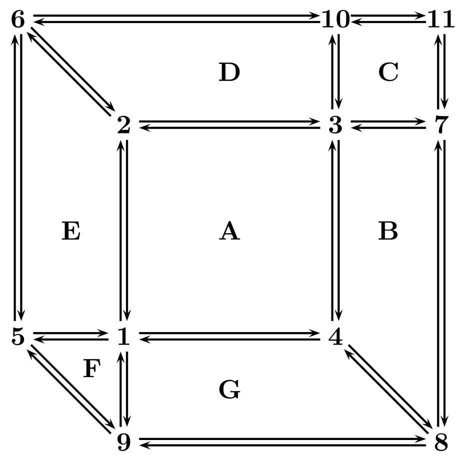
Coedge	pair	next	prev	head	tail	face
AB						
AC						
AD						
BA						
BC						
BD						
CA						
CB						
CD						
DA						
DB						
DC						

vertex	coedge
1	
2	
3	
4	

face	coedge
A	
B	
C	
D	

Problem 7

Write the routine `void Get_Ring(std::vector<face*>& ring, const face* f);`. Given a face `f`, it should fill `ring` with a list of all faces that share a vertex with `f`. The faces in `ring` should be listed in counterclockwise order, but it does not matter which face is listed first. For example, if `f`=A in the diagram below, then `ring`=(B,C,D,E,F,G) would be an acceptable output. Note that the faces need not be triangles. This is not the same algorithm as presented in class; that algorithm only returns faces that share an edge and would return (B,D,E,F,G). Your routine should be written in something close to C++ syntax. *Although it is not required, you are encouraged to implement this. It will help you understand the structure and its traversal better. It will also help you debug your algorithm. You will be asked to devise an algorithm to perform a task on this structure on the final. It will be a task you have not seen before in this class. Be ready for it.*


6
10
11
D
C
2
3
7
E
A
B
5
1
4
F
G
9
8