1. ***Chasing Puppies.*** Watch *Chasing Puppies*[1], the Ferran Hurtado Memorial Lecture at the 32nd Canadian Conference on Computational Geometry by Jeff Erickson. (The video is about ***one hour long***. Make sure you start watching it early on.)

   Let γ be a generic simple closed curve in the plane (the trail), and let h and p be the positions of the human and the puppy on the trail respectively. The human and the puppy always stay on the trail; in other words, points h and p themselves can be view as a map from the parameter space $\mathbb{S}^1$ to the plane $\mathbb{R}^2$. The puppy wants to find its way to the human, by simply running towards the direction of the trail that reduces its distance to the human. The goal of the human is to move back-and-forth on the trail, in order to guide the puppy back. Can the puppy always catch the human?

   To solve the problem, Jeff defined the *puppy* and *human* (hooman?) diagrams in his talk. The ***puppy diagram*** is the subset of *critical* points in the human-puppy configuration space (which is a torus); in other words, the puppy diagram is the set

   $$\left\{(h,p) \in \mathbb{S}^1 \times \mathbb{S}^1 : \partial D(h,p)/\partial p = 0\right\},$$

   where $\boldsymbol{D(h,p)}$ is the distance between the human and the puppy.

   The ***human diagram*** is the subset of points at which the human can be in the puppy vision space (which is an infinite cylinder); in other words, the human diagram is the set

   $$\left\{(p, L(h,p)) \in \mathbb{S}^1 \times \mathbb{R} : (h,p) \text{ is critical}\right\},$$

   where $\boldsymbol{L(h,p)}$ is the (signed) distance between h and the tangent line of γ at p.

   Consider the map F from the human-puppy configuration space to the puppy vision space, by sending (h,p) to $(p, L(h,p))$. Prove the following statements:

   (a) Map F is a homeomorphism between the puppy and human diagrams.

   (b) Map F sends a contractible cycle in the puppy diagram to a contractible cycle in the human diagram.

   (c) Map F sends an essential (that is, non-contractible) cycle in the puppy diagram to an essential cycle in the human diagram.

   (d) The puppy diagram has only one essential closed curve besides the diagonal. (Yes, I know Jeff has already proved this in the talk. But do you understand the argument?)

[1] https://www.youtube.com/watch?v=Ysk0yhO4jVk

2. ***Rolling cube puzzles.*** A *rolling cube puzzle*[2] consists of a ***cube***, sitting on top of a ***map*** assembled by a collection of unit-size squares identifying along the edges; at most two squares meet at a common edge at a time. The map naturally identifies with a surface, possibly with boundaries. (For sake of simplicity, you can assume the surface is orientable.) In addition, for purpose of the problem, each face of the cube is identical in size to the map squares; also a *unique* color is associated with each face of the cube.

   An ***instance*** of the rolling cube puzzle on a given map is a pair of squares, each with a *color label*. A rolling cube puzzle instance is ***solvable*** if one can transfer the cube from the starting position to the ending position by rolling the cubes: First, put the cube at the starting position with the color required by the label facing up. Roll the cube square-by-square along a common intersecting edge as axis; the cube is never allowed to go off the map. At the ending position, the color of the cube facing up has to match the color required by the label. Notice that the order of the colors on the side of the cube does not matter (in particular, you have the freedom to choose how the cube is initially placed as long as the top of the cube matches the color label.)

   Design and analyze an algorithm, which after some preprocessing on the input map, solves an instance of the rolling cube puzzle efficiently. To get full credit, your algorithm must spend time linear to the size of the map in preprocessing, and constant time per instance afterwards. *[Hint: Why is this problem here?]*

[2] see, for example, commercial games like *Rubek* and *Hocus* or free games like *Bloxorz*