6-7. The value of conjectures

1. Find a link diagram satisfying the following conditions. You may use SnapPy to draw them.

	Alternating	Reduced	Connected
i)	Yes	Yes	Yes
ii)	Yes	Yes	No
iii)	Yes	No	Yes
iv)	Yes	No	No
v)	No	Yes	Yes
vi)	No	Yes	No
vii)	No	No	Yes
viii)	No	No	No

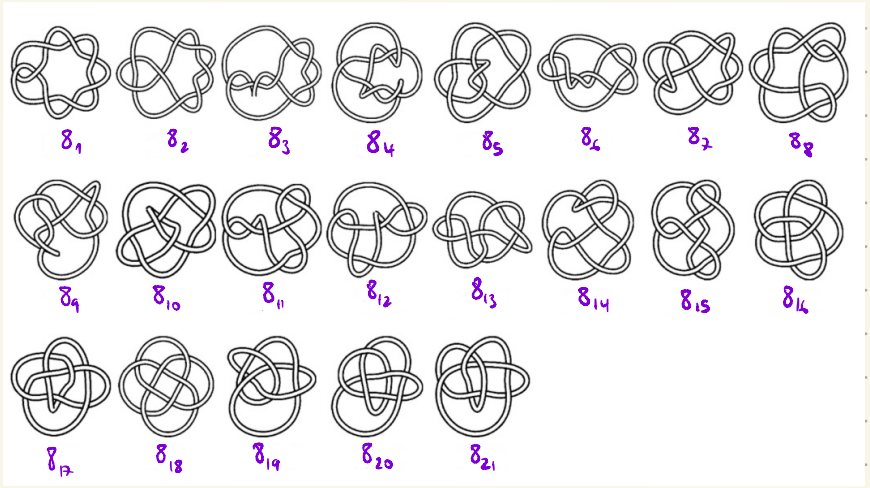
A link diagram is alternating if following each strand's path, overcrossings and undercrossings alternate.

A link diagram D is reduced if we cannot find two arcs such that

D = 

Connected: a link diagram is connected if ignoring (under/over) crossings, the diagram is connected.

2. Identify the three nonalternating prime knots with 8 crossings:



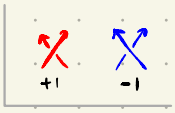
Hint: start from the end.

3. Choose a prime knot with $n \leq 7$ crossings and use Tait's first conjecture to prove that its crossing number is n.

Tait's first conjecture: Any reduced alternating link diagram has the smallest number of crossings

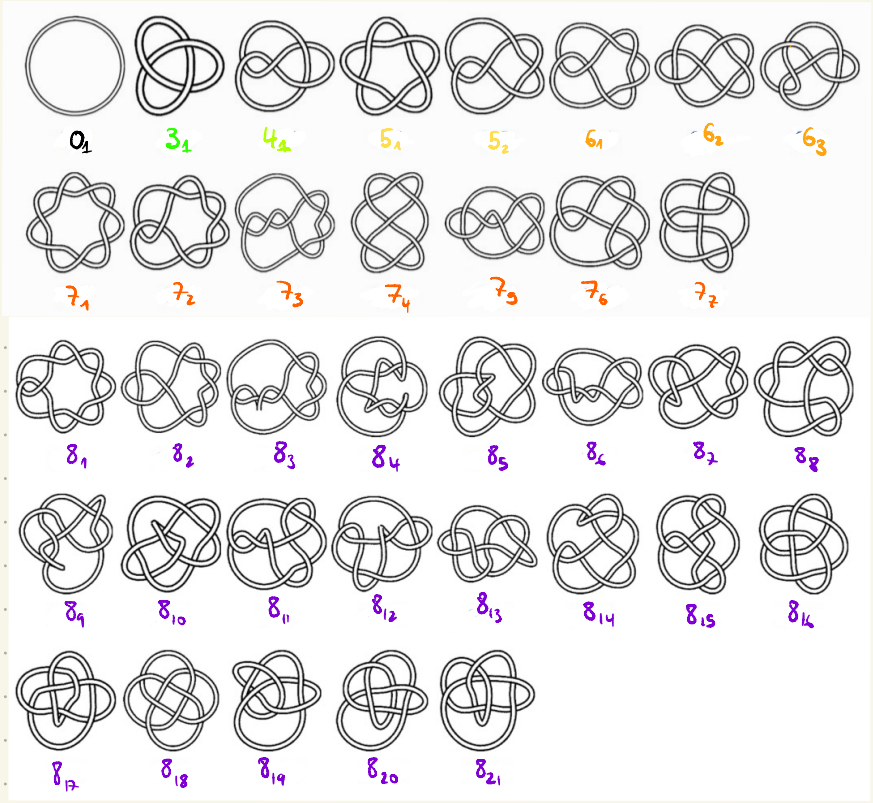
Tait's second conjecture: Any two reduced reduced alternating connected diagrams for the same link have the same writhe.

4. Let D be an oriented link diagram and let D' be its mirror image. You may use SnapPy.



Explore how $wr(D)$ and $wr(D')$ are related. Conjecture a relation between them. Prove your conjecture.

5. Let D be a reduced alternating connected diagram of an amphichiral knot. Use Tait's second conjecture and the previous exercise to show that $D \neq 5_1$. Prove more generally that the number of crossings in D must be even.

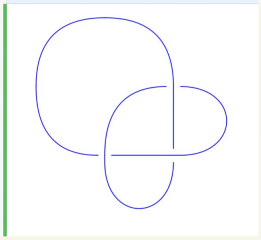


Game: Choose a prime knot, tangle it as much as you can (within reason) using SnapPy, and save it. Then share it on the chat so that the other group can open it in SnapPy's editor. First one to identify the knot wins. You may use any technique from the course.

Some useful commands:

- Basic example:

```
import snappy
PD= [(1, 5, 2, 4), (5, 3, 6, 2), (3, 1, 4, 6)]
L_snappy = snappy.Link(PD)
L=L_snappy.sage_link()
L.plot()
```



- `L.is_alternating()` returns whether the diagram is alternating. (dependent on the diagram)
- `L.writhe()` returns the writhe of the diagram (invariant of alternating diagrams ONLY)
- `L.is_colorable(n)` returns whether the diagram is n-colorable (independent of the diagram)
- `L.jones_polynomial()` returns the Jones polynomial (independent of the diagram)