Pair Programming

1 What?

Pair programming is exactly what is sounds like: "two people working together at a single computer" (`www.extremeprogramming.org/`). The practice was originally popularized by a software development methodology called Extreme Programming (XP), and its effects have been studied by a number of researchers. Anecdotally, I used it in my first programming module and have never looked back.

2 Why?

It may be the case that some of you already know how to program, and either don't think you need extra help or don't want to hold someone else's hand while they learn. Regardless, here are a few reasons, supported by research [1, 2], why we are doing pair programming and why you should be excited about it:

- By explaining your ideas to another person, you will become better at articulating your thoughts.
- You will benefit from the insights of your peers and benefit from sharing your ideas with them.
- You will write better code.
- You will spend less time frustrated.
- You will be better prepared for more complex software engineering, either in future modules or in a job, where collaboration is not optional (both in the sense that you won't be able to do everything yourself, and your employer won't let you).

3 Who?

Given the motivations stated above, it is useful to engage in pair programming with as many other people as possible, as the way you explain things might make sense to one person but not another. Please try to rotate partners throughout the course of the module.

Ultimately, you and your partner will be collectively responsible for the code you produce: there is no "my part" and "your part," and it is not acceptable for one person to do all of the work and then add their partner's name. If your partner is unwilling to help or is otherwise disengaged, please contact the PGTA or lecturer and we will do our best to help resolve the situation.

4 How?

The division of labour in pair programming is split between the *driver* and the *navigator*. The driver is responsible for carrying out the actions of the pair (typing, moving the mouse, etc.), and the navigator is responsible for both the higher-level design decisions (does this part fit with the rest of the codebase?) and for reviewing the driver's work (is there a bug?).

Some people may naturally prefer (or be better at) one role over the other, but to minimise frustrations and keep pair programming fun, it is essential to switch roles throughout. This should be done roughly every 15 minutes, although you can wait for a natural breaking point as long as no one stays in the same role for more than 20 minutes.

5 Where?

Pair programming will be used throughout the module, primarily in labs and in the project. In labs, pair programming is required and will be strictly enforced. In the project, it is encouraged, although we do not expect that it will be followed quite as closely as in labs (e.g., pairs may work out ideas first together and then contribute code separately, rather than always work together on the same computer), and each partner must produce their own report separately.

References

[1] Jennifer Bevan, Linda Werner, and Charlie McDowell, Guidelines for the use of pair programming in a freshman programming class, In *Proceedings of the 15th Conference on Software Engineering Education and Training* (CSEET'02), 2002.

[2] Laurie Williams, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller, In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):192–212, 2002.