Lab Assignment

Problem 1

To implement `(i-like-more?)`, the method was declared in `(make-person)` so it can access `preference-list`. Two additional helper methods were defined:

1. `(present?)`: This method simply checks if a target exists in a given list. It's used to if `person1` and/or `person2` exist within the context of a person's `preference-list`.
2. `(distance-from-start)`: This method counts how far down a person is on the subjects's `preference-list`. It is zero indexed, and a lower number is considered to be more preferred
3. `(currently-unengaged)` operates on a simple filter on whether the list is filled with single people. That is, `(filter single? people)`.
4. `(couple?)` and `(single?)` are very similar methods in terms of implementation, as they check if a person is involved with someone else. However, the two are separate methods because the two are semantically different — `(single?)` checks if a only one person is single, but `(couple?)` checks if two people are couple.

Because `(make-person)` is being constantly redefined, odd bugs result. When new people are created, the people creation code is moved to the bottom of the file.

Problem 2

An issue I encountered is that in the second `(if)` block `(if (and (not (single? me)) ((me 'i-like-more?) proposer (me 'intended))) ... )` is that `(write-line '(buzz-off-creep2))` gets called even the couple is declaring their undying love for one another. It's not that obnoxious of a bug, so it stays.

Problem 3

I implemented the `(dialouge)` procedure from the lunar lander to write out into the console what each person is saying/thinking. It prints out the state, and the the person who called it (via `(me 'name)`).

Problem 4

Let's assume that **not** everybody is engaged to someone else at the end of the stable marriage procedure. But the stable marriage procedure creates couples from all the participants. Therefore, the stable marriage algorithm generates stable pairs.

Problem 5

When P is proposing, P starts proposing to people at the top its preference list. If P is rejected, P moves down their preference list, decreasing in person preference each time.

When A is being proposed to, A would only change their current romantic partner only if they like the proposer more. There is no reason for someone to leave their current partner unless they like the new proposer more.

Problem 6

Prove that the marriages produced by the stable marriage algorithm are indeed stable. Suppose not: then let m and w be a man and woman not matched together. Show m and w could not both wish to run off together. Hint: show m had to have proposed to w at some time: where did their love go wrong?

In the initialization, each person listed all their preferences for other people. If `m` and `n` don't end up together at the end, then they must've proposed at some point, as either `m` or `n` would need to go through their entire preference once.

Problem 7

There was no difference the generated couples when `(match-make men women)` and `(match-make women men)` when run with the original 12 people. But with the new people (Bob, Carol, Ted, Alice), who proposes to who (that is, men or women) matters, as when men propose to women, Bob ends up with Carol, and Ted with Alice. But when women propose to men Carol ends up with Ted, and Alice with Bob. In other words: **the proposers get their first choice**.

As for the question regarding whether I'd like to propose marriage or get proposed to, I'm going to sidestep that entirely. I'd rather have a large enough potential pool of spouses where it wouldn't be practical to make a list of top preferences, and instead spend time learning and becoming close with my spouse. At least I have my romantic life figured out, unlike this class.