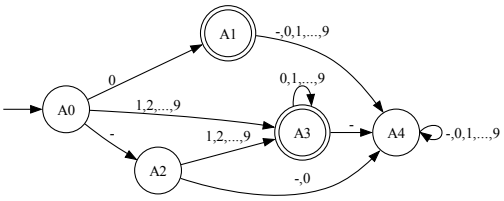
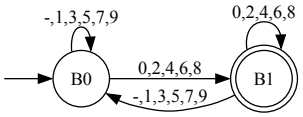
MAU22C00 Assignment 9, Due Friday 1 December 2023
Solutions

1. Give a strongly deterministic finite state automaton which recognises the language of *even* integers.
   Hint: This is the kind of problem where a bit of extra time thinking about different ways to solve the problem can end up saving you a lot of time overall.
   Solution: You're probably familiar with the fact that the even integers are precisely those whose last digit is even. So a string of characters is an even integer if it is an integer and the last character is a `0`, `2`, `4`, `6`, or `8`. In other words, the language we're looking for is the intersection of the language of integers, for which I've already given you a strongly deterministic finite state automaton, and the language of strings whose last character is a `0`, `2`, `4`, `6`, or `8`, for which we can easily construct a strongly deterministic finite state automaton. Once we have those we can construct a strongly deterministic finite state automaton for the intersection by the product construction.

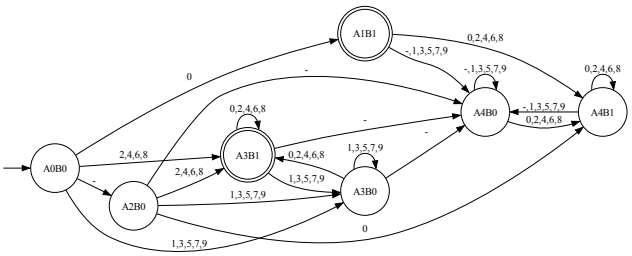
   Now that we have a plan it's fairly easy to carry it through. The finite state automaton from the integers was



   A simple finite state automaton which recognises strings ending with `0`, `2`, `4`, `6`, or `8` is



   A finite state automaton for the intersection is



You could put in the states A0B1, A1B0 and A2B1 if you wanted to, but they're not reachable so there's no point. The finite state automaton above is not quite the simplest possible. If you construct one via the Myhill-Nerode Theorem, for example, you'll get one where the states A4B0 and A4B1 are combined, but you'll have to do significantly more work to get it.

2. Construct a regular expression which describes the language of non-empty strings of `0`'s and `1`'s with exactly as many occurences of the substring `01` as of the substring `10`.
   Hint: The description I've just given of this language is not the simplest description possible. You will find this problem easier if you can find a simpler description for it.
   Solution: Go through the string and count the number of `01`'s minus the number of `10`'s you've seen. Initially this difference is 0. The first character is either `0` or `1`. If it's 0 then the difference will jump to 1 the first time we read a `1` and then go back 0 the next time we read a `0`, and continue oscillating between 0 and 1 as we read more characters. It's final value will be 0 if the final character is `0` and 1 if the final character is `1`. Having exactly as many `01`'s as `10`'s therefore means that if the first character is `0` then so is the last character. The situation if the first character is a `1` is similar, but now the difference oscilates between 0 and -1. The conclusion we get in this case is that having exactly as many `01`'s as `10`'s therefore means that if the first character is `0` then so is the last character. So in general, a string belongs to this language if and only if its first character and last character are the same.

   Once you've found this alternate characterisation of the language it's simple to write a regular expression which is nearly correct. We either have a `0`, then any number of `0`'s and `1`'s, and then a `0` or we have a `1`, then any number of `0`'s and `1`'s, and then a `1`. Any number of `0`'s and `1`'s is matched by the regular expression `(0|1)*` so the two possibilities are `0(0|1)*0` and `1(0|1)*1` and the union of these is `(0(0|1)*0)|(1(0|1)*1)`. This is very nearly correct. The problem is that it fails to match the strings `0` and `1`, which are in the language. We need to add those in by hand, obtaining `0|1|(0(0|1)*0)|(1(0|1)*1)`.

3. Show that the language of palindromes is not regular.
   Hint: You can do this using either the Pumping Lemma or the Myhill Nerode Theorem. In this case I think the Pumping Lemma is a bit easier.
   Solution: Suppose the language is regular. Then it has a pumping length p. Choose two distinct letters, say `x` and `y` and let w be p `x`'s followed by a `y` and then another p `x`'s. This is a member of the language. The Pumping Lemma says we can write this as a concatentation abc where b is a non-empty substring from the first p characters and ab^nc is a member of the language for each natural number n. a must consist solely of `x`'s so if we take $n = 0$ we get a string with fewer than p `x`'s followed by a single `y` and then p `x`'s. This is not a member of the language, so we have a contradiction. The assumption that the language is regular is therefore false.