Sheet 11

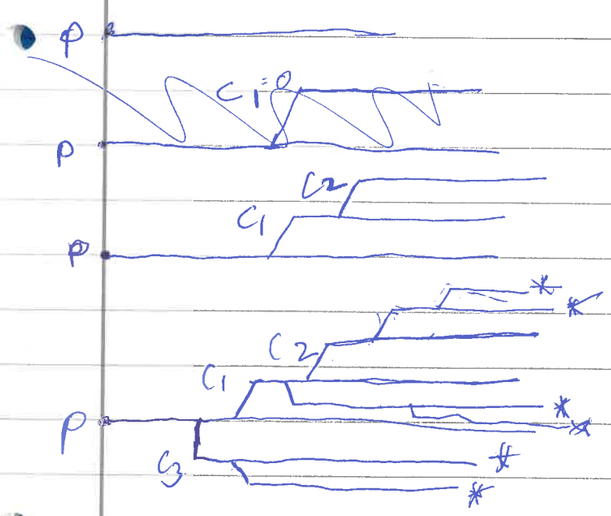
11-1

```
int
{
    for(; argc > 1; argc--) {
        if (0 == fork()) {
            (void) fork();
        }
    }
    return
}
```

① ./foo argc is 1 never enters for loop

② ./foo a argc = 2 [argv[0] = 0 so one iteration
argv[1] = a 2-1 = 1
2 args

③ ./foo a b argc = 3 [argv[0] = 0 3-1 = 2 ① } iter.
argv[1] = a 2-1 = 1 ① }
argv[2] = b
3 args



1 ① ./foo

3 ② ./foo a

9 ③ ./foo a b

27 ./foo a b c

81 ./foo a b c d

So # of child processes is 81 - 1 = 80

11.2

4 * 2 * x + 7

a) <digit> * [(<digit> * <var>) + <digit>]

<constant> * [(<digit> * <var>) + <digit>]

<factor> * [(<digit> * <var>) + <digit>]

<term> * [(<digit> * <var>) + <digit>]

<term> * [(<const> * <factor>) + <constant>]

<term> * [(<factor> * <factor>) + <factor>]

<term> * [(<term> * factor) + <term>]

<term> * [<expression> + <term>]

<term> * (<expression>)

<term> * <factor>

<term>

<expression> //

b) Yes, the associativity of the multiplication so basically adding the extra brackets eg. (2 * x), to make it clear that we are multiplying first instead of adding then multiplying.