

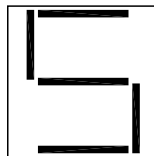
**2a:** Recall from problem 1e that  $\binom{n}{2}$  is the number of pairs possible among  $n$  points. Phrased differently, if  $n$  people are at a party, and everyone shakes hands with everyone else, then  $\binom{n}{2}$  handshakes will occur at the party.

Suppose, now, that you did not know the formula for  $\binom{n}{2}$ . Let's count the handshakes a different way. Each time a person arrives at the party, the person shakes hands with everyone else who was already there. Use this description to write a recursive program to compute  $\binom{n}{2}$ .

**2b:** The following definition makes use of `quotient`, which is a procedure built in to Scheme. The `quotient` procedure divides one integer by another, but returns only the quotient, ignoring the remainder. For example, `(quotient 2718 10)` evaluates to 271, because 271 is the number of times 10 goes into 2718. Your task is to list the `(round-down 2718)`, using the same format as at the bottom of page 26 in your textbook.

```
(define round-down
  (lambda (n)
    (if (< n 10)
        n
        (* (round-down (quotient n 10))
           10))))
```

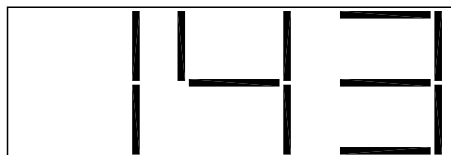
**2c:** [HKK 2.23] Suppose we have made images for each of the digits 0–9, which we name `zero-bb`, `one-bb`, ..., `nine-bb`. For example, if you evaluate `five-bb`, you get the following image:



You can load up images of the digits using,

```
(load "~mc27/public/digits.scm")
```

- (a) Write a procedure `image-of-digit` that takes a single parameter  $d$  that is an integer satisfying  $0 \leq d \leq 9$  and returns the image corresponding to  $d$ . You should definitely use a `cond`, because you would otherwise have to nest the `ifs` ridiculously deep.
- (b) Using the procedure `image-of-digit`, write another procedure `image-of-number` that takes a single parameter  $n$  that is a nonnegative integer and returns the image corresponding to it. Thus, `(image-of-number 143)` would return the following image:



*Hint:* Use the Scheme procedures `quotient` and `remainder` to break  $n$  apart. Also, you may use the procedure `side-by-side` from Exercise ?? without redefining it here.

**2d:** [HKK 2.18] Prove by induction that for every nonnegative integer  $n$  the following procedure computes  $2n$ :

```
(define f
  (lambda (n)
    (if (= n 0)
        0
        (+ 2 (f (- n 1))))))
```

**2e:** [HKK 2.20] Prove that the following procedure computes  $n/(n+1)$  for any nonnegative integer  $n$ . That is, (f  $n$ ) computes  $n/(n+1)$  for any integer  $n \geq 0$ .

```
(define f
  (lambda (n)
    (if (= n 0)
        0
        (+ (f (- n 1))
            (/ 1 (* n (+ n 1)))))))
```