CS 135 — Racket Quick Notes

Website: student.cs.uwaterloo.ca/~cs135

Prefix Notation

```
(* (/ 6 2) (+ 1 2))
```

Racket uses **prefix notation**. A substitution step evaluates the leftmost / innermost sub-expression (i.e., the one that closes first).

Arithmetic Operators

```
+ - \* /
```

These operators can take **two or more arguments** and associate left-to-right.

Negation (unary -):

```
(- 144) ; = -144
(- -21) ; = 21
```

This is an **error** (needs ≥ 2 args for +): (+ 72)

Practice Questions

```
(+ 3 (* 5 2))
(/ (- 10 4) 2)
(* 7 (+ 3 1))
(- (- (- 10 100)) (- -10 -15) (- -20))
```

Number Types (CS 135)

- **Nat** — natural numbers (includes 0)
- **Int** — integers (includes Nat)
- **Rat** — rationals (includes Int)

Useful Functions

```
(quotient 43 7)   ; Int → takes Ints only
(remainder 43 7)  ; Nat → takes Ints only

(max 1 2 3 4) ; = 4
(min 1 2 3 4) ; = 1
(max 9)       ; = 9
(min 9)       ; = 9

(sqr 12)      ; = 144
(expt 3 4)    ; = 81  ; 3^4
```

Constant Definitions

Constants cannot be redefined after their initial definition.

```
(define x 7)
(define y (+ x 4)) ; x = 7, y = 11
```

Example:

racket

```
(define cost-per-pizza 12)
(define cost-per-drink 3)
(define pizzas-needed 8)
(define drinks-needed 15)

(+ (* cost-per-pizza pizzas-needed)
   (* cost-per-drink drinks-needed))
```

Boolean Expressions

- Relational operators: `<` `>` `<=` `>=` `=` ⟶ return **Bool** (`#true` or `#false`).
- Logical operators: `and`, `or`, `not`.

Short-circuit evaluation

- `and` ⟶ returns `#false` immediately if any argument is false.
- `or` ⟶ returns `#true` immediately if any argument is true.

(DML/design recipe ideas can help keep expressions simple and readable.)
Example:

```
(+ (= 6 7) 8) ; false (0) + 8 = 8
```