

# Contents

Practice Questions . . . . .	1
Number Types (CS 135) . . . . .	1
Useful Functions . . . . .	1
Constant Definitions . . . . .	1
Boolean Expressions . . . . .	2

Website: <https://student.cs.uwaterloo.ca/~cs135/> ### Prefix Notation

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

Racket uses **prefix notation** A substitution step finds the leftmost/inner most subexpression, i.e. the first closing bracket ### Arithmetic Operators

`+ - * /`

can operate on **two or more values** linearly from the leftmost argument

## Negation

`(- 144) ; = -144`

`(- -21) ; = 21`

This is an **error**:

`(+ 72)`

## Practice Questions

`(+ 3 (* 5 2))`

`(/ (- 10 4) 2)`

`(* 7 (+ 3 1))`

`(- (- (- 10 100)) (- -10 -15) (- -20))`

## Number Types (CS 135)

1) **Nat** = Natural numbers (0 included)

2) **Int** = Integers (includes Nat)

3) **Rat** = Rational numbers (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 (34)`

## Constant Definitions

**Constants cannot be redefined** after the initial definitions

`(define x 7)`

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

Examples:

```
(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

- Operators: `<` `>` `<=` `>=` `=` `→` return a **Bool** (true or false).
- Logical operators: **and**, **or**, **not** `###` Short-circuit evaluation
- **and** `→` false if any argument is false.
- **or** `→` true if any argument is true.

DML can help to make keep statements simpler, more elegant, and more readable.

Examples:

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