

MORE PRACTICE WITH RACKET 0.2

COMPUTER SCIENCE 61AS

The Basics of Racket

1. What notation does Racket use and what are the benefits of using it?

Solution: Racket uses prefix notation which means that the operator is put at the beginning of the expression and followed by its arguments. For example: `(+ 3 4)`

This allows the user to nest expressions within each other and for mathematics operations that can take more than one argument to do so without needing another sign. For example:

```
(+ (* 4 3) (- 4 2)) -> 14
(+ 3 4 5 6) -> 18
```

2. What do we mean by 'functional programming'?

Solution: Functional programming is the idea that we can express a variety of computational algorithms by using the value returned by one function as an argument to another function.

Words and Sentences with Racket

What will the following expressions return?

1. `(first '(hello there))`

Solution: `hello`

2. `(bf '(hello there))`

Solution: `(there)` Note how the word `(there)` is in parentheses. This means that it is a sentence even if it is a one word sentence.

3. `(define (polite sentence) (sentence 'please sentence))`
`(polite '(go to the mall))`

Solution: `error`

Booleans, Predicates and Special Forms

1. What are booleans?

Solution: Booleans are a data type with only two possible values true or false denoted by `#t` and `#f`, respectively.

2. What are predicates?

Solution: Predicates are functions that return a true or false value.

3. Why does `new-if` not work exactly the same as `if`?

```
(define (new-if predicate if-true if-false)
  (if predicate if-true if-false))
```

Solution: `new-if` doesn't work because it is not a special form. Unlike the normal `if`, `new-if` will evaluate all of the arguments you present it with; therefore, it will evaluate both the `if-true` and `if-false` value which is not what you want.

4. What do the following expressions evaluate to?

- a. `(= (+ 2 2) 5)`

Solution: `#f`

- b. `(if 'happy`
 `'(i am happy)`
 `(/ 1 0))`

Solution: `'(i am happy)`

c. `(equal? 'there (bf '(hello there)))`

Solution: #f. Remember, `bf` returns a sentence with everything but the first word. This means that `(bf '(hello there))` returns a one word sentence `(there)`. This is not equal to the word `'there`.

5. Write a procedure `num-name` that takes in single digit numbers and outputs the word equivalent. Example: `(num-name 3)` Returns: `three`

Solution:

```
(define (num-name number)
  (cond ((= number 0) 'zero)
        ((= number 1) 'one)
        ((= number 2) 'two)
        ((= number 3) 'three)
        ((= number 4) 'four)
        ((= number 5) 'five)
        ((= number 6) 'six)
        ((= number 7) 'seven)
        ((= number 8) 'eight)
        ((= number 9) 'nine)))
```

Domain and Range

Domain and range in CS are just like domain and range from math. The domain of a procedure is the type(s) of arguments that it can accept and the range of a procedure is the type of the output.

1. What is the domain and range of `last`?

Solution: domain - word or sentence; range - word

2. What is the domain and range of `+`?

Solution: domain - number, number, number...; range - number

3. Find and fix the bugs in the following code which finds the color of a card. Cards are represented as a sentence with their value as the first word and the suit as the second word. For example, the Jack of Hearts is '(jack hearts) and 9 of Spades is '(nine spades)

```
(define (card-color sent)
  (if (or (= (last sent) 'hearts)
          (= (last sent) 'diamonds))
      'red
      'black))
```

Solution: The domain of = is a number. Because the (last sent) is a word and not a number, the above code is not correct. The = should be replaced with equal?.