

Fundamentos - Problemas

Marco A L Barbosa

malbarbo.pro.br

Os exercícios sem referências estão licenciados com uma Licença Creative Commons - Atribuição-CompartilhaIgual 4.0 Internacional.



<https://github.com/malbarbo/na-progfun>

- 1) [sicp 1.1] Dado o seguinte programa, qual é o resultado impresso pelo Racket em resposta a cada expressão? Assuma que as expressões serão avaliadas na ordem em que são apresentadas.

```
10
```

```
(+ 5 3 4)
```

```
(- 9 1)
```

```
(- 9)
```

```
(/ 6 2)
```

```
(/ 4)
```

```
(*)
```

```
(+)
```

```
(+ (* 2 4) (- 4 6))
```

```
(define a 3)
```

```
(define b (+ a 1))
```

```
(+ a b (* a b))
```

```
(= a b)
```

```
(if (and (> b a) (< b (* a b)))  
    b  
    a)
```

```
(cond [(= a 4) 6]  
      [(= b 4) (+ 6 7 a)]  
      [else 25])
```

```
(+ 2 (if (> b a) b a))
```

```
(* (cond [(> a b) a]  
         [(< a b) b]  
         [else -1])  
   (+ a 1))
```

- 2) Qual das seguintes construções formam expressões sintaticamente válidas em Racket? Justifique.

- a. `34 + 45`
- b. `(aplicar * 1 3 4 5)`
- c. `((if (> 0 8) - /) 10 20)`
- d. `(+ - * /)`
- e. `(if (> 4 9) 10)`

3) [sicp 1.2] Traduza a seguinte expressão para a forma prefixa

$$\frac{5 + 4 + (2 - (3 - (6 + \frac{4}{5})))}{3(6 - 2)(2 - 7)}$$

- 4) [tspl 2.2.2] Experimente os procedimentos `+`, `-`, `*` e `/` e determine as regras do Racket para o tipo do valor de retorno para cada procedimento quando são dados diferentes tipos de argumentos numéricos.
- 5) [sicp 1.4] O modelo de avaliação visto em sala permite combinações em que os operadores são expressões compostas. Use esta observação para descrever o comportamento do seguinte procedimento:

```
(define (a-plus-abs-b a b)
  ((if (> b 0) + -) a b))
```

6) [sicp 1.5] Ben Bitdiddle inventou um método para determinar se um interpretador está usando avaliação com ordem aplicativa ou avaliação com ordem normal. Ele definiu os seguintes procedimentos:

```
(define (p) (p))
```

```
(define (test x y)
  (if (= x 0)
      0
      y))
```

Então avaliou a seguinte expressão

```
(test 0 (p))
```

Qual é o comportamento que Ben irá observar com um interpretador que usa avaliação com ordem aplicativa? Qual é o comportamento que ele irá observar com um interpretador que usa avaliação com ordem normal? Explique a sua resposta.

7) Faça uma função chamada `tres-digitos?` que recebe um número natural `n` e verifica se `n` tem exatamente 3 dígitos. Não use `if` nem `cond`. Confira na janela de interações se a função funciona de acordo com os exemplos a seguir

```
> (tres-digitos? 99)
#f
> (tres-digitos? 100)
#t
> (tres-digitos? 999)
#t
> (tres-digitos? 1000)
#f
```

8) Faça uma função chamada `traco-meio?` que recebe uma string `s` e verifica se o caractere no meio de `s` é `"-"`. Não use `if` nem `cond`. Confira na janela de interações se a função funciona de acordo com os exemplos a seguir

```
> (traco-meio? "lero-lero")
#t
> (traco-meio? "quase-meio")
#f
> (traco-meio? "-")
#t
```

- 9) Faça uma função chamada `adiciona-ponto` que recebe um string `frase` e adiciona um ponto final na `frase` se ela ainda não tiver um. Confira na janela de interações se a função funciona de acordo com os exemplos a seguir

```
> (adiciona-ponto "Vou contar")
"Vou contar."
> (adiciona-ponto "Corri.")
"Corri."
```

- 10) Faça uma função chamada `ordem` que recebe três números distintos, `a`, `b` e `c` e determina se a sequência `a`, `b`, `c` está em ordem crescente, decrescente ou não está em ordem. Confira na janela de interações se a função funciona de acordo com os exemplos a seguir

```
> (ordem 3 8 12)
" crescente "
> (ordem 3 1 4)
" sem ordem "
> (ordem 3 1 0)
" decrescente "
```

Referências

- [sicp]. Structure and Interpretation of Computer Programs
- [tspl]. The Scheme Programming Language