Programação Funcional
Fundamentos

Marco A L Barbosa malbarbo.pro.br

Começando

1) O que é um literal?
2) O que é uma função primitiva?
3) O que é uma expressão?
4) O que significa avaliar uma expressão?
5) O que é uma combinação?
6) Como é chamado a expressão mais a esquerda de uma combinação? E as demais expressões?
7) Qual é a regra de avaliação para uma combinação?
8) Qual é o propósito de uma definição?
9) O que é uma função?
10) O que é uma forma especial?
11) Por que o `and` e o `or` são formas especiais e o `not` não é?
12) A ordem que as expressão de uma combinação são avaliadas pode alterar o valor da combinação? Explique.
13) Quando devemos utilizar o operador =?
14) Qual a diferença entre as funções `eq?` e `equal?`?
15) O que é uma definição com autorreferência?
16) O que é um processo recursivo?

Praticando

17) Escreva a expressão $(3 + 1) \times 8/(4 - 1)$ em Racket.
18) Qual o resultado da avaliação da expressão `(* 3 (+ 1 2) (/ 10 2) 2)`? Mostre o passo a passo da avaliação.
19) Qual o resultado da avaliação de `(3 + 8)` na janela de interações do Racket? E de `3 + 8`. Explique.

Na implementação dos exercício a seguir use apenas as funções presentes no material "Resumo da linguagem Racket" disponível na página da disciplina.

20) Faça uma função chamada `area-retangulo` que recebe dois argumentos, a `largura` e a `altura` de um retângulo, e calcula a sua área. Confira na janela de interações se a função funciona de acordo com os exemplos a seguir

```
> (area-retangulo 3 5)
15
```

```
> (area-retangulo 2.0 2.5)
5.0
```

21) Faça uma função chamada `produto-anterior-posterior` que recebe um número inteiro `n` e calcula o produto de `n`, `n + 1` e `n - 1`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

```
> (produto-anterior-posterior 3)
24
> (produto-anterior-posterior 1)
0
> (produto-anterior-posterior -2)
-6
```

22) Faça uma função chamada `so-primeira-maiuscula` que recebe uma palavra não vazia (string) como parâmetro e crie uma nova string convertendo a primeira letra da palavra para maiúscula e o restante da palavra para minúscula. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

```
> (so-primeira-maiuscula "paula")
"Paula"
> (so-primeira-maiuscula "ALFREDO")
"Alfredo"
```

23) Faça uma função chamada `exclamacao` que recebe dois argumento, uma string `frase` e um número natural `n`, e produz a mesma frase adicionando `n` pontos de exclamação no final da frase. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

```
> (exclamacao "Nossa" 3)
"Nossa!!!"
> (exclamacao "Que legal" 1)
"Que legal!"
> (exclamacao "Nada" 0)
"Nada"
```

24) Faça uma função chamada `par?` que recebe um número natural `n` e indica se `n` é par. Um número é par se o resto da divisão dele por 2 é igual a zero. Não use `if` nem `cond` e nem a função pré-definida `even?`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

```
> (par? 3)
#f
> (par? 6)
#t
```

25) 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`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

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

26) Faça uma função `maximo` que encontre o máximo entre dois números dados. Não use a função pré-definida `max`. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

```
> (maximo 3 5)
5
> (maximo 8 4)
8
> (maximo 6 6)
6
```

27) Faça uma função chamada `ordem` que recebe três números distintos, `a`, `b` e `c` e determina se a sequencia `a, b, c` está em ordem crescente, decrescente ou não está em ordem. Use os operadores relacionas com três argumentos. Use o método de substituição para verificar se a sua função funciona corretamente de acordo com os exemplos a seguir. Em seguida confira a respostas na janela de iterações do DrRacket.

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

28) [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))
```

29) [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.