

PL1-Racket

Início



Disciplina: AARE Paradigmas de Linguagens de Programação

Professor: Prof. Ausberto S. Castro V.

E-mail: ascv@uenf.br

Data: 9 de setembro de 2021

Nome Completo: João Vítor Fernandes Dias

Data: 9 de setembro de 2021

Total Exercícios Resolvidos: Todos

Exercícios

Arquivo 01-primeiro.rkt

Primeiro programa em Racket

1. Execute o programa e indicar o que faz cada linha do código fonte do programa. Quais funções estão definidas nas 4 linhas. Explique cada uma delas.

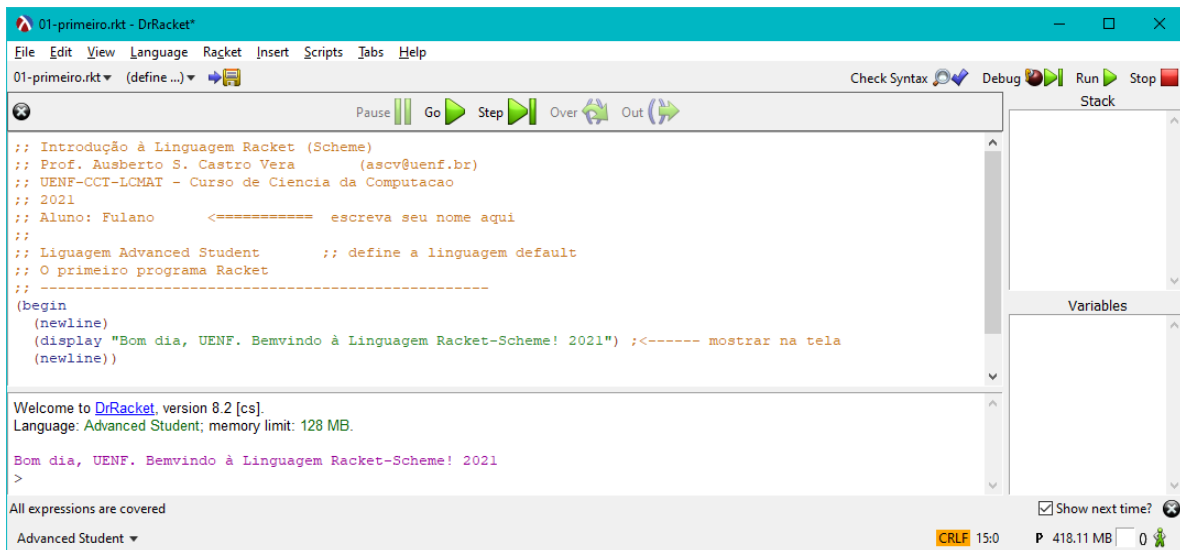
Imagens Ausberto

```
;; Introdução à Linguagem Racket (Scheme)
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; 2021
;; Aluno: Fulano      <===== escreva seu nome aqui
;;
;; Linguagem Advanced Student      ;; define a linguagem default
;; O primeiro programa Racket
;; -----
(begin
  (newline)
  (display "Bom dia, UENF. Bemvindo à Linguagem Racket-Scheme! 2021") ;<----- mostrar na tela
  (newline))

Welcome to DrRacket, version 8.2 [cs].
Language: Advanced Student; memory limit: 128 MB.

Bom dia, UENF. Bemvindo à Linguagem Racket-Scheme! 2021
>
```

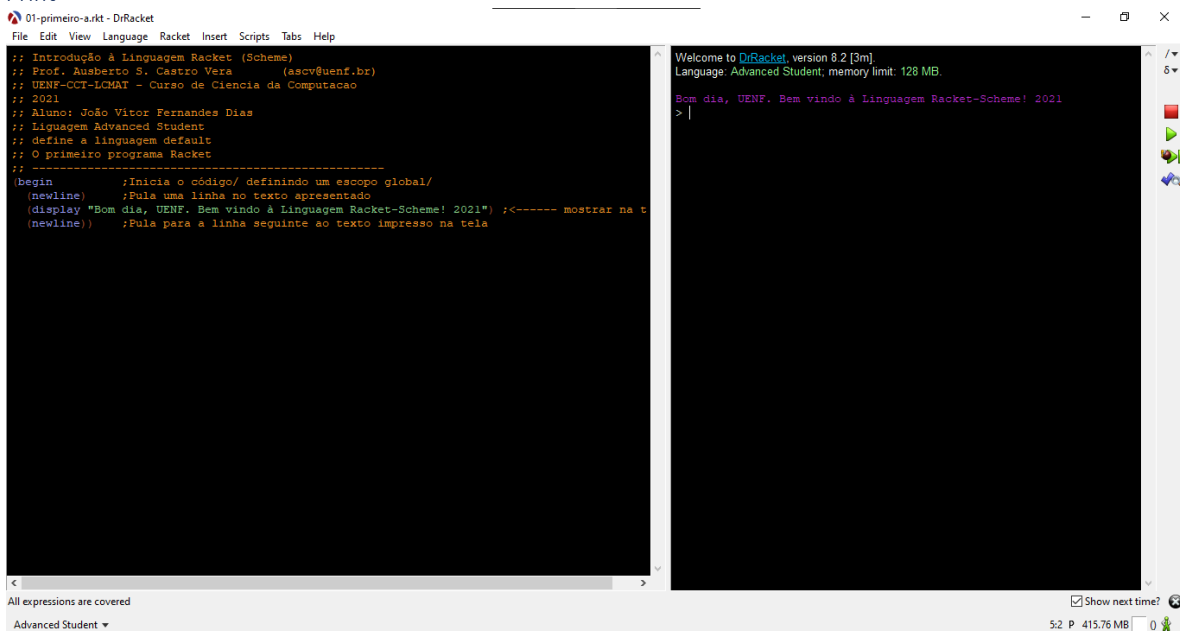
Código fonte



Interações (shell)

Explicar aqui:

Print



Código:

```

(begin      ;Inicia o código/ definindo um escopo global/
  (newline) ;Pula uma linha no texto apresentado
  (display "Bom dia, UENF. Bem vindo à Linguagem Racket-Scheme! 2021") ;<----- mostrar na tela
  (newline)) ;Pula para a linha seguinte ao texto impresso na tela

```

2. Agregar linhas de código para mostrar na parte executável, a mensagem “Pratica 01 – Linguagem Racket”, o nome completo do aluno e a data atual

Print

The screenshot shows the DrRacket IDE with a file named "01-primeiro-b.rkt". The editor contains a Racket script that prints a welcome message, the student's name, and the current date. The output window on the right shows the execution results, which match the script's output.

```
;; Introdução à Linguagem Racket (Scheme)
;; Prof. Auberio S. Castro Vera (ascv@uenf.br)
;; UENF-CCIT-LICMAT - Curso de Ciencia da Computacao
;; 2021
;; Aluno: João Vitor Fernandes Dias
;; Linguagem Advanced Student
;; define a linguagem default
;; O primeiro programa Racket
-----
(begin
  (newline) ;Inicia o código/ definindo um escopo global
  (display "Bom dia, UENF. Bem vindo à Linguagem Racket-Scheme! 2021") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "Pratica 01 - Linguagem Racket") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "João Vitor Fernandes Dias") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "02/09/2021") ;mostra na tela
  (newline) ;Pula para a linha seguinte ao texto impresso na tela
)
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student; memory limit: 128 MB.

Bom dia, UENF. Bem vindo à Linguagem Racket-Scheme! 2021
Pratica 01 - Linguagem Racket
João Vitor Fernandes Dias
02/09/2021
>
```

Código

```
(begin      ;Inicia o código/ definindo um escopo global
  (newline) ;Pula uma linha no texto apresentado
  (display "Bom dia, UENF. Bem vindo à Linguagem Racket-Scheme! 2021") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "Pratica 01 – Linguagem Racket") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "João Vítor Fernandes Dias") ;mostra na tela
  (newline) ;Pula uma linha no texto apresentado
  (display "02/09/2021") ;mostra na tela
  (newline) ;Pula para a linha seguinte ao texto impresso na tela
) ;Termina o begin
```

3. Execute o programa e mostre os resultados

Print

```

;; Introdução à Linguagem Racket (Scheme)
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciência da Computação
;; 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
;;
;; -----
;; 1) Instrução
;;
;; (display " UENF-CCT-LCMAT-CC, 2021")
;; (newline)
;; (display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)")
;; (newline)
;; (display " Aluno: João Vitor Fernandes Dias ")
;; (newline)
;;
;;
;; Números e aritmética
;; -----
;;
;; (display "Soma 23 + 28 = ")      (+ 23 28)
;; (display "Produto 14*17 = ")    (* 14 17)
;; (display "Combinando 5 + (3*7) = ") (+ 5 (* 3 7))
;; (display "Combinando (2 + (3*4))/2 - 4 = ") (- (/ (+ 2 (* 3 4)) 2) 4)
;; (display "Raiz quadrada de 4= ") (sqrt 4)
;; (display "Raiz quadrada de 2= ") (sqrt 2)
;; (display "Complexos - raiz quadrada de -1= ") (sqrt -1)
;;
;; #i significa "inexato"
;; (newline)
;; (display "Valor de Pi+1 ")      (+ pi 1)
;; (display "Seno 90 graus: ")    (sin (/ pi 2))
;; (display "Cosseno 60 graus: ") (cos (/ pi 3))
;; (display "Cosseno 45 graus: ") (cos (/ pi 4))
;; (display "Logaritmo Natural de 15: ") (log 15)
;; (display "exponente 2^3 = ")    (expt 2 3)
;; (display "exponente 4^(1/2) = ") (expt 4 1/2)
;; (display "Maximo de 1 3 4 2 3 = ") (max 1 3 4 2 3)
;; (display "minimo de 1 3 4 2 3 = ") (min 1 3 4 2 3)

```

```

;; Introdução à Linguagem Racket (Scheme)
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciência da Computação
;; 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
;;
;; -----
;; 2) Seguindo a instrução
;;
;; (display " UENF-CCT-LCMAT-CC, 2021")
;; (display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)")
;; (display " Aluno: João Vitor Fernandes Dias ")
;;
;; -----
;; Números e aritmética-----
;;
;; (display "Soma 23 + 28 = ")      (+ 23 28)
;; (display "Produto 14*17 = ")    (* 14 17)
;; (display "Combinando 5 + (3*7) = ") (+ 5 (* 3 7))
;; (display "Combinando (2 + (3*4))/2 - 4 = ") (- (/ (+ 2 (* 3 4)) 2) 4)
;; (display "Raiz quadrada de 4= ") (sqrt 4)
;; (display "Raiz quadrada de 2= ") (sqrt 2)
;; (display "Complexos - raiz quadrada de -1= ") (sqrt -1) ;; #i significa "inexato"
;; (display "Valor de Pi+1 ")      (+ pi 1)
;; (display "Seno 90 graus: ")    (sin (/ pi 2))
;; (display "Cosseno 60 graus: ") (cos (/ pi 3))
;; (display "Cosseno 45 graus: ") (cos (/ pi 4))
;; (display "Logaritmo Natural de 15: ") (log 15)
;; (display "exponente 2^3 = ")    (expt 2 3)
;; (display "exponente 4^(1/2) = ") (expt 4 1/2)
;; (display "Maximo de 1 3 4 2 3 = ") (max 1 3 4 2 3)
;; (display "minimo de 1 3 4 2 3 = ") (min 1 3 4 2 3)
;; (display "valor absoluto de 3 = ") (abs 3)
;; (display "valor absoluto de -4 = ") (abs -4) (newline)
;;
;; Expressões "quote": listas de objetos tratados como dados
;; "quotes obriga as listas serem tratadas como DADOS"
;; (quote (2 4 6))
;; (quote (/ 4 (* 3 7)))
;; (list 1 2 3 4) ;; comentario
;; ((a b) (3 5))
;; (+ 2 (* 5 7)) (newline)

```

Código:

```

;; -----Números e aritmética-----
(display "Soma 23 + 28 = ")      (+ 23 28)
(display "Produto 14*17 = ")    (* 14 17)
(display "Combinando 5 + (3*7) = ") (+ 5 (* 3 7))
(display "Combinando (2 + (3*4))/2 - 4 = ") (- (/ (+ 2 (* 3 4)) 2) 4)
(display "Raiz quadrada de 4= ") (sqrt 4)
(display "Raiz quadrada de 2= ") (sqrt 2)
(display "Complexos - raiz quadrada de -1= ") (sqrt -1) ;; #i significa "inexato"
(display "Valor de Pi+1 ")      (+ pi 1)

```

```
(display "Seno 90 graus: ") (sin (/ pi 2))
(display "Coseno 60 graus: ") (cos (/ pi 3))
(display "Coseno 45 graus: ") (cos (/ pi 4))
(display "Logaritmo Natural de 15: ") (log 15)
(display "exponente 2^3 = ") (expt 2 3)
(display "exponente 4^(1/2) = ") (expt 4 1/2)
(display "Maximo de 1 3 4 2 3 = ") (max 1 3 4 2 3)
(display "minimo de 1 3 4 2 3 = ") (min 1 3 4 2 3)
(display "valor absoluto de 3 = ") (abs 3)
(display "valor absoluto de -4 = ") (abs -4) (newline)
```

; Expressões "quote": listas de objetos tratados como dados
 ""quotes obriga as listas serem tratadas como DADOS"

```
(quote ( 2 4 6))
(quote (/ 4 (* 3 7)))
'(1 2 3 4) ;; comentario
'((a b)(3 5))
'(+ 2 (* 5 7)) (newline)
```

4. Escreva programas Racket para as seguintes expressões:

4.1. $H = (4 - (7^2 + 6^3) / 3) - (6 + (5 - (2^4 - 8)))$

Print

The screenshot shows the DrRacket IDE with a file named '02-numeros-buht - DrRacket'. The editor contains Scheme code for problem 4.1. The output window on the right shows the results of the execution.

```
;; Introdução à Linguagem Racket (Scheme)
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LQMAI - Curso de Ciência da Computação
;; 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
#lang racket
(display " UENF-CCT-LQMAI-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)" (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; -----Numeros e aritmetica-----
(display "Cálculo 4.1") (newline)

(display "Direto: ")
(- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8))))

(display "Com display: ")
(display (- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8)))))
(newline)

(display "Definindo o H previamente: ")
(define H (- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8)))))
(display H)
```

The output window shows the following results:

```
Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student; memory limit: 128 MB.
UENF-CCT-LQMAI-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Cálculo 4.1
Direto: -87.3
Com display: -262/3
Definindo o H previamente: -262/3
>|
```

Código:

```
(display "Cálculo 4.1") (newline)
```

```
(display "Direto: ")
(- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8)))))
```

```
(display "Com display: ")
(display (- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8)))))
(newline)
```

```
(display "Definindo o H previamente: ")
(define H (- (- 4 (/ (+ (expt 7 2) (expt 6 3)) 3)) (+ 6 (- 5 (- (expt 2 4) 8)))))
```

(display H)

4.2. Escreva um NOVO programa Racket que calcule o valor da expressão:

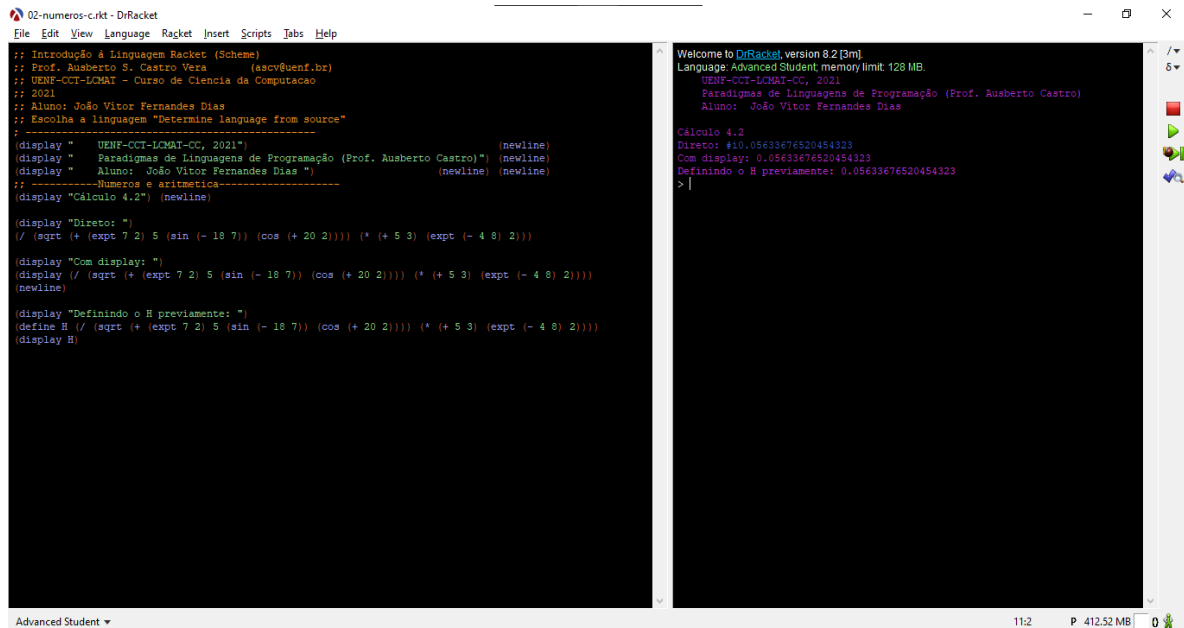
Expressão

$$\frac{\sqrt{7^2 + 5 + \sin(18 - 7) + \cos(20 + 2)}}{(5 + 3) * (4 - 8)^2}$$

`sqrt (7^2+5+sin (18-7) +cos (20+2)) / ((5+3) *(4-8)^2)`

`=` `sqrt (7^2+5+sin (18-7) +cos (20+2)) / ((5+3) *(4-8)^2)`
`0.0563367652`

Print



```
02-numeros-c.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução 4 Linguagem Racket (Scheme)
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;-----
(display "Cálculo 4.2") (newline)

(display "Direto: ")
(/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2)))

(display "Com display: ")
(display (/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2))))
(newline)

(display "Definindo o H previamente: ")
(define H (/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2))))
(display H)
```

Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
Cálculo 4.2
Direto: #10.05633676520454323
Com display: 0.05633676520454323
Definindo o H previamente: 0.05633676520454323
>|

Código:

(display "Cálculo 4.2") (newline)

(display "Direto: ")

(/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2)))

(display "Com display: ")

(display (/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2))))
(newline)

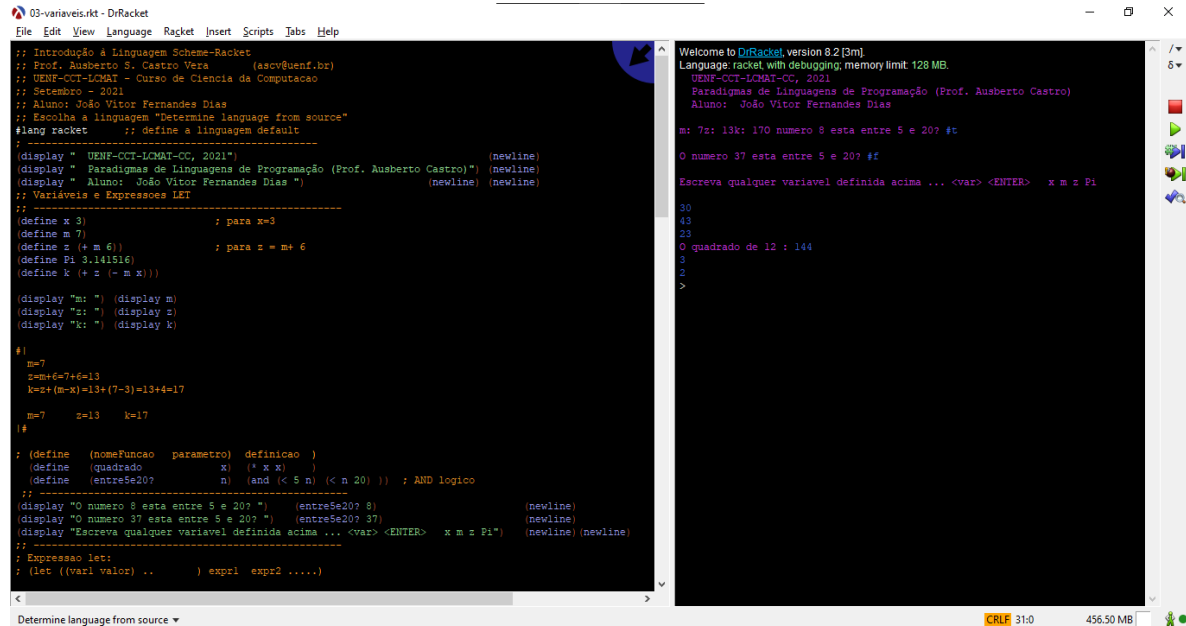
(display "Definindo o H previamente: ")

(define H (/ (sqrt (+ (expt 7 2) 5 (sin (- 18 7)) (cos (+ 20 2)))) (* (+ 5 3) (expt (- 4 8) 2))))
(display H)

Arquivo 03-variaveis.rkt

5. Execute o programa e indique o valor das variáveis m, z, k

5.1. Print



The screenshot shows the DrRacket IDE with a file named '03-variaveis.rkt'. The editor displays a Racket program that defines variables m, z, and k, and prints their values. The output window on the right shows the results of the program execution.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciência da Computação
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
#lang racket      ;; define a linguagem default

;; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; Variáveis e Expressões LET
;; -----
(define x 3)          ; para x=3
(define m 7)
(define z (+ m 6))    ; para z = m+ 6
(define Pi 3.141516)
(define k (+ z (- m x)))

(display "m: ") (display m)
(display "z: ") (display z)
(display "k: ") (display k)

#|
m=7
z=m+6=7+6=13
k=z+(m-x)=13+(7-3)=13+4=17

m=7   z=13   k=17
|#

; (define (nomeFuncao parametro) definicao )
; (define (quadrado x) (* x x) )
; (define (entre5e20? n) (and (< 5 n) (< n 20) )) ; AND logico
;; -----
(display "O numero 8 esta entre 5 e 20? ") (entre5e20? 8) (newline)
(display "O numero 37 esta entre 5 e 20? ") (entre5e20? 37) (newline)
(display "Escreva qualquer variavel definida acima ... <var> <ENTER> x m = Pi") (newline) (newline)
;; Expressão let:
; (let ((var1 valor) .. ) expr1 expr2 .....)

Welcome to DrRacket, version 8.2 [3m]
Language: racket with debugging, memory limit 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

m: 7z: 13k: 17O numero 8 esta entre 5 e 20? #t
O numero 37 esta entre 5 e 20? #f
Escreva qualquer variavel definida acima ... <var> <ENTER> x m = Pi
30
43
23
O quadrado de 12 : 144
3
2
>
```

Código:

;; Variáveis e Expressões LET

;; -----

(define x 3) ; para x=3

(define m 7)

(define z (+ m 6)) ; para z = m+ 6

(define Pi 3.141516)

(define k (+ z (- m x)))

(display "m: ") (display m)

(display "z: ") (display z)

(display "k: ") (display k)

#|

m=7

z=m+6=7+6=13

k=z+(m-x)=13+(7-3)=13+4=17

m=7 z=13 k=17

#

; (define (nomeFuncao parametro) definicao)

(define (quadrado x) (* x x))

(define (entre5e20? n) (and (< 5 n) (< n 20))) ; AND logico

;; -----

(display "O numero 8 esta entre 5 e 20? ") (entre5e20? 8) (newline)

(display "O numero 37 esta entre 5 e 20? ") (entre5e20? 37) (newline)

```
(display "Escreva qualquer variavel definida acima ... <var> <ENTER> x m z Pi") (newline)(newline)
```

```
;; -----
```

```
; Expressao let:
```

```
; (let ((var1 valor) .. ) expr1 expr2 .....
```

```
(let ((x 24) ) (+ x 6) )
```

```
#|
```

Permite que o X tenha o valor 24
dentro da expressão "+ x 6",
assim retornando o valor 24+6=30

```
|#
```

```
(let ((a 5) (b 8) ) (+ 3 (* a b)) );
```

```
#|
```

Permite que o a tenha o valor 5 e que b tenha o valor 8
dentro da expressão "+ 3 (* a b)",
assim retornando o valor 3+5*8=43

```
|#
```

```
(let ((op1 +) (op2 *) (x 5) ) (op1 3 (op2 4 x)) )
```

```
#|
```

Permite que o op1 tenha o valor "+", op2 tenha o valor "*" e x tenha o valor 5
dentro da expressão "(op1 3 (op2 4 x))",
obtendo então a expressão "+ 3 (* 4 5)",
assim retornando o valor 3+4*5 = 23

```
|#
```

```
(display "O quadrado de 12 : ")(quadrado 12)
```

```
(let ((a 1) (b 2)) (+ a b))
```

```
#|
```

Define o valor de a como sendo 1 e o valor de b como sendo 2
Então faz a soma de a com b
Retornando o valor a+b=1+2=3

```
|#
```

```
(let ((b 6) (c 8)) (- c b))
```

```
#|
```

Define o valor de b como sendo 6 e o valor de c como sendo 8
Então faz a subtração de b em c
Retornando o valor c-b=8-6=2

```
|#
```

```
#| ;Tentativa de Bhaskara
```

```
(let ((a 1) (b 3) (c 5))
```

```
(
```

```
(let ((vezes *) (mais +) (divide /) (subtrai -) (quatro 4) (dois 2))
```

```
(
```

```
(let ( (delta (subtrai (vezes b b) (vezes quatro a c) )))
```

```
(
```

```
( let ((raizDelta (sqrt delta)) (menosB (- 0 b)) (doisA (vezes dois a)))
```

```
(
```

```
(let ((menosBsobreDoisA (divide menosB doisA)) (raizDeltasobreDoisA (divide raizDelta doisA)))
```

```
(
```


5.2. Valores das variáveis m=7; z=13; k=17

m=7
z=m+6=7+6=13
k=z+(m-x)=13+(7-3)=13+4=17

5.3. Explicar o significado de cada uma das 3 expressões de iteração let

Print

```
(let ((x 24) ) (+ x 6) )
#|
Permite que o X tenha o valor 24
dentro da expressão "+ x 6",
assim retornando o valor 24+6=30
|#

(let ((a 5) (b 8) ) (+ 3 (* a b)) );
#|
Permite que o a tenha o valor 5 e que b tenha o valor 8
dentro da expressão "+ 3 (* a b)",
assim retornando o valor 3+5*8=43
|#

(let ((op1 +) (op2 *) (x 5) ) (op1 3 (op2 4 x)) )
#|
Permite que o op1 tenha o valor "+", op2 tenha o valor "*" e x tenha o valor 5
dentro da expressão "(op1 3 (op2 4 x))",
obtendo então a expressão "+ 3 (* 4 5)",
assim retornando o valor 3+4*5 = 23
|#

(display "O quadrado de 12 : ") (quadrado 12)

(let ((a 1) (b 2)) (+ a b))
#|
Define o valor de a como sendo 1 e o valor de b como sendo 2
Então faz a soma de a com b
Retornando o valor a+b=1+2=3
|#

(let ((b 6) (c 8)) (- c b))
#|
Define o valor de b como sendo 6 e o valor de c como sendo 8
Então faz a subtração de b em c
Retornando o valor c-b=8-6=2
|#
```

Explicações:

```
(let ((x 24) ) (+ x 6) )
#|
Permite que o X tenha o valor 24
dentro da expressão "+ x 6",
assim retornando o valor 24+6=30
|#
```

```
(let ((a 5) (b 8) ) (+ 3 (* a b)) );
#|
Permite que o a tenha o valor 5 e que b tenha o valor 8
dentro da expressão "+ 3 (* a b)",
assim retornando o valor 3+5*8=43
|#
```

```
(let ((op1 +) (op2 *) (x 5)) (op1 3 (op2 4 x)))
```

#|
 Permite que o op1 tenha o valor "+", op2 tenha o valor "*" e x tenha o valor 5 dentro da expressão "(op1 3 (op2 4 x))", obtendo então a expressão "+ 3 (* 4 5)", assim retornando o valor $3+4*5 = 23$

#

```
(display "O quadrado de 12 : ")(quadrado 12)
```

```
(let ((a 1) (b 2)) (+ a b))
```

#|
 Define o valor de a como sendo 1 e o valor de b como sendo 2
 Então faz a soma de a com b
 Retornando o valor $a+b=1+2=3$

#

```
(let ((b 6) (c 8)) (- c b))
```

#|
 Define o valor de b como sendo 6 e o valor de c como sendo 8
 Então faz a subtração de b em c
 Retornando o valor $c-b=8-6=2$

#

6. Escreva 2 expressões do tipo let e explique o seu significado

Primeira Tentativa – Bhaskara – Não consegui

Print

The screenshot shows the DrRacket IDE with a Racket script on the left and the output on the right. The script defines a function for the Bhaskara formula and tests it with various inputs. The output shows the results of these tests, including the calculation of the discriminant and the roots of the equation.

```
#| ;Tentativa de Bhaskara
(let ((a 1) (b 3) (c 5))
  (
    (let ((vezes *) (mais +) (divide /) (subtrai -) (quatro 4) (dois 2))
      (let ( (delta (subtrai (vezes b b) (vezes quatro a c) )))
        (let ((raizDelta (sqrt delta)) (menosB (- 0 b)) (doisA (vezes dois a)))
          (let ((menosBsobreDoisA (divide menosB doisA)) (raizDeltasobreDoisA (divide raizDelta
            (let ((X1 (mais menosBsobreDoisA raizDeltasobreDoisA)) (X2 (subtrai menosBsobreDoisA
              (let ((teste1 0))
                (+ X1 0)
              ))
            ))
          ))
        (let ((teste2 0))
          (+ X2 0)
        ))
      ))
    )
  )
#|
```

Output:

```
Welcome to DrRacket, version 8.2 [3m]
Language: racket with debugging, memory limit 128 MB.
UNF-CCT-ICMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

m: 7s: 13k: 170 numero 8 esta entre 5 e 20? #t
O numero 37 esta entre 5 e 20? #f

Escreva qualquer variavel definida acima ... <var> <ENTER> x m z Pi
30
43
23
O quadrado de 12 : 144
3
2
>
```

Código

```
#| ;Tentativa de Bhaskara
(let ((a 1) (b 3) (c 5))
  (
```

```

(let ((vezes *) (mais +) (divide /) (subtrai -) (quatro 4) (dois 2))
  (
    (let ( (delta (subtrai (vezes b b) (vezes quatro a c) )))
      (
        (let ((raizDelta (sqrt delta)) (menosB (- 0 b)) (doisA (vezes dois a))))
          (
            (let ((menosBsobre doisA (divide menosB doisA)) (raizDeltasobre doisA (divide raizDelta doisA)))
              (
                (let ((X1 (mais menosBsobre doisA raizDeltasobre doisA)) (X2 (subtrai menosBsobre doisA
raizDeltasobre doisA)))
                  (
                    (let ((teste1 0))
                      (
                        + X1 0
                      )
                    )
                  )
                )
              )
            )
          )
        )
      )
    )
  )
)

```

|#

Segunda tentativa – Mais simples

Print

The screenshot shows the DrRacket IDE with a Racket script on the left and its output on the right. The script defines several arithmetic operations and a function to solve a quadratic equation. The output shows the results of these operations and the execution of the quadratic equation solver.

```

03-variaveis.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

#|
Permite que o a tenha o valor 5 e que b tenha o valor 8
dentro da expressão "+ 3 (* a b)",
assim retornando o valor 3+5*8=43
|#
(let ((op1 +) (op2 *) (x 5) (op1 3 (op2 4 x)) )
  #|
Permite que o op1 tenha o valor "+", op2 tenha o valor "*" e x tenha o valor 5
dentro da expressão "(op1 3 (op2 4 x))",
obtido então a expressão "+ 3 (* 4 5)",
assim retornando o valor 3+4*5 = 23
|#
(display "O quadrado de 12 : ") (quadrado 12)
(let ((a 1) (b 2)) (+ a b))
#|
Define o valor de a como sendo 1 e o valor de b como sendo 2
Então faz a soma de a com b
Retornando o valor a+b=1+2=3
|#
(let ((b 6) (c 8)) (- c b))
#|
Define o valor de b como sendo 6 e o valor de c como sendo 8
Então faz a subtração de b em c
Retornando o valor c-b=8-6=2
|#
#|Tentativa de Bhaskara
(let ((a 1) (b 3) (c 5))
  (
    (let ((vezes *) (mais +) (divide /) (subtrai -) (quatro 4) (dois 2))
      (
        (let ( (delta (subtrai (vezes b b) (vezes quatro a c) )))
          (
            (let ((raizDelta (sqrt delta)) (menosB (- 0 b)) (doisA (vezes dois a))))
              (
                (let ((menosBsobre doisA (divide menosB doisA)) (raizDeltasobre doisA (divide raizDelta doisA)))
                  (
                    (let ((X1 (mais menosBsobre doisA raizDeltasobre doisA)) (X2 (subtrai menosBsobre doisA
raizDeltasobre doisA)))
                      (
                        + X1 0
                      )
                    )
                  )
                )
              )
            )
          )
        )
      )
    )
  )
)

```

Output:

```

Welcome to DrRacket, version 8.2 [3m].
Language: racket with debugging, memory limit: 128 MB.
UENF-CCT-ICMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

m: 7z: 13k: 170 numero 8 esta entre 5 e 20? #t
O numero 37 esta entre 5 e 20? #f
Escreva qualquer variavel definida acima ... <var> <ENTER> x m = Pi
30
43
23
O quadrado de 12 : 144
3
2
>

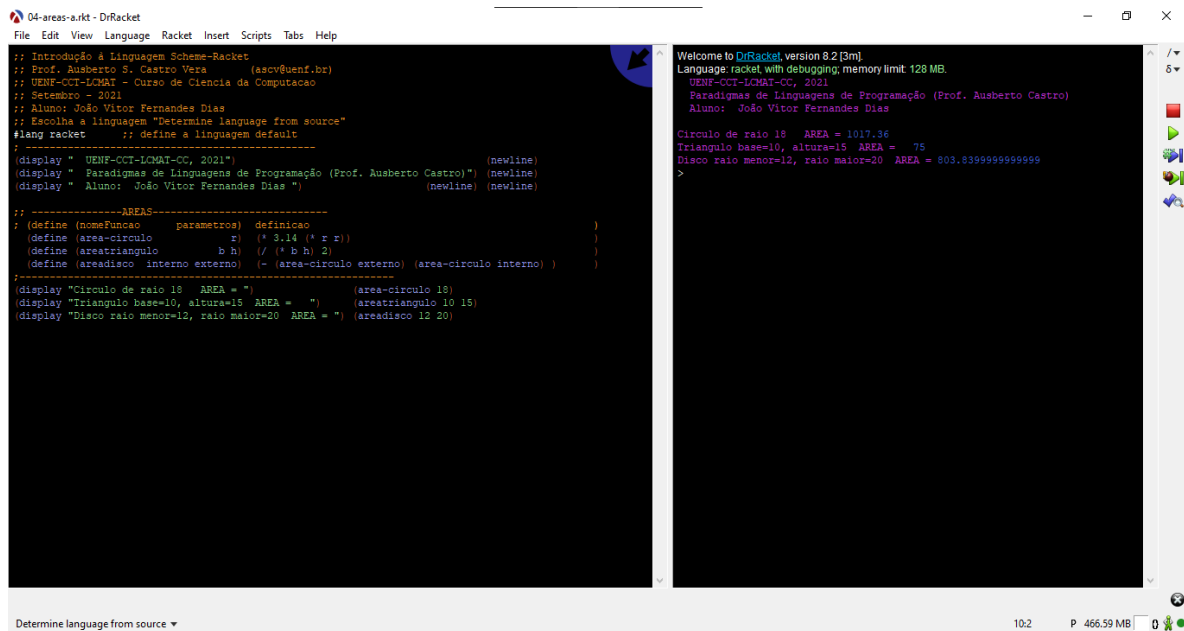
```

Código

```
(let ((a 1) (b 2)) (+ a b))  
#|  
Define o valor de a como sendo 1 e o valor de b como sendo 2  
Então faz a soma de a com b  
Retornando o valor a+b=1+2=3  
|#  
  
(let ((b 6) (c 8)) (- c b))  
#|  
Define o valor de b como sendo 6 e o valor de c como sendo 8  
Então faz a subtração de b em c  
Retornando o valor c-b=8-6=2  
|#
```

Arquivo 04-areas.rkt

7. Execute o programa e indique o que faz o programa Print



```
;; Introdução à Linguagem Scheme-Racket  
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)  
;; UENF-OCT-LOGMAT - Curso de Ciência da Computação  
;; Setembro - 2021  
;; Aluno: João Vitor Fernandes Dias  
;; Escolha a linguagem "Determine language from source"  
#lang racket      ;; define a linguagem default  
;  
; -----AREAS-----  
; (define (nomeFuncao parametros) definicao)  
; (define (area-circulo r) (* 3.14 (* r r)))  
; (define (areatriangulo b h) (/ (* b h) 2))  
; (define (areadisco interno externo) (- (area-circulo externo) (area-circulo interno)))  
;  
(display "Circulo de raio 18 AREA = ") (area-circulo 18)  
(display "Triangulo base=10, altura=15 AREA = ") (areatriangulo 10 15)  
(display "Disco raio menor=12, raio maior=20 AREA = ") (areadisco 12 20)
```

```
Welcome to DrRacket, version 8.2 [3m].  
Language: racket with debugging, memory limit 128 MB.  
UENF-OCT-LOGMAT-OC, 2021  
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)  
Aluno: João Vitor Fernandes Dias  
  
Circulo de raio 18 AREA = 1017.36  
Triangulo base=10, altura=15 AREA = 75  
Disco raio menor=12, raio maior=20 AREA = 803.8399999999999  
>
```

Explicação

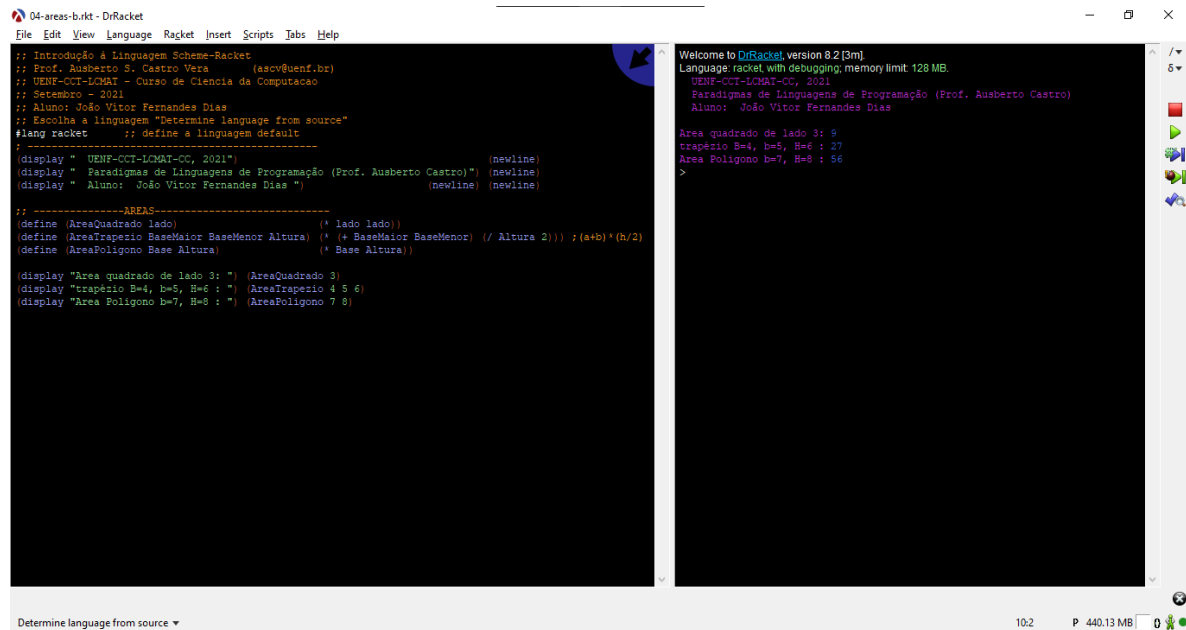
Esse código define fórmulas para cálculo de áreas dos planos: Círculo, triângulo, disco. Abaixo das definições, elas são testadas com alguns valores.

Código

```
;; -----AREAS-----  
; (define (nomeFuncao parametros) definicao)  
; (define (area-circulo r) (* 3.14 (* r r)))  
; (define (areatriangulo b h) (/ (* b h) 2))  
; (define (areadisco interno externo) (- (area-circulo externo) (area-circulo interno)))  
;  
(display "Circulo de raio 18 AREA = ") (area-circulo 18)  
(display "Triangulo base=10, altura=15 AREA = ") (areatriangulo 10 15)  
(display "Disco raio menor=12, raio maior=20 AREA = ") (areadisco 12 20)
```

8. Escreva um programa Racket para calcular a área de um quadrado qualquer, a área de um trapézio e a área de um polígono.

Print



The screenshot shows the DrRacket IDE interface. The left pane contains a Racket script with the following code:

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-ICMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
#lang racket
;; define a linguagem default
;-----
(display " UENF-CCT-ICMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;-----
;ABAS
(define (AreaQuadrado lado) (* lado lado))
(define (AreaTrapezio BaseMaior BaseMenor Altura) (* (+ BaseMaior BaseMenor) (/ Altura 2))) ;(a+b)*(h/2)
(define (AreaPoligono Base Altura) (* Base Altura))

(display "Area quadrado de lado 3: ") (AreaQuadrado 3)
(display "trapézio B=4, b=5, H=6 : ") (AreaTrapezio 4 5 6)
(display "Area Poligono b=7, H=8 : ") (AreaPoligono 7 8)
```

The right pane shows the output of the program:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket with debugging, memory limit 128 MB.
UENF-CCT-ICMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Area quadrado de lado 3: 9
trapézio B=4, b=5, H=6 : 27
Area Poligono b=7, H=8 : 56
>
```

The status bar at the bottom indicates "Determine language from source", "10.2", and "P 440.13 MB".

Código

```
(define (AreaQuadrado lado) (* lado lado))
(define (AreaTrapezio BaseMaior BaseMenor Altura) (* (+ BaseMaior BaseMenor) (/ Altura 2))) ;(a+b)*(h/2)
(define (AreaPoligono Base Altura) (* Base Altura))

(display "Area quadrado de lado 3: ") (AreaQuadrado 3)
(display "trapézio B=4, b=5, H=6 : ") (AreaTrapezio 4 5 6)
(display "Area Poligono b=7, H=8 : ") (AreaPoligono 7 8)
```

9. fórmula $V = \pi R^2 A$, onde as variáveis V, R e A representam, respectivamente, o volume, o raio e a altura
- Print

The screenshot shows the DrRacket IDE with a file named '04-areas-c.rkt'. The editor contains a Racket script that sets up a language environment, defines a function to calculate the volume of a cylinder, and displays the result. The output window on the right shows the welcome message and the execution results, including the volume calculation.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem "Determine language from source"
#lang racket ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;-----
#|
Escreva um programa NOVO completo para calcular o volume de um galão de óleo
utilizando a fórmula V = pi*R^2*A, onde as variáveis V, R e A representam,
respectivamente, o volume, o raio e a altura
|#
(define (VolumeOleo R A) (* pi R R A ))
(display "Volume do barril de óleo de raio 2 e altura 4: ")
(define V (VolumeOleo 2 4))
(display V)
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit 128 MB.
Module Language: only a module expression is allowed, either
#lang <language-name>
or
(module <name> <language> ...)
in: racket
Interactions disabled.
```

Código

```
(define (VolumeOleo R A) (* pi R R A ))
(display "Volume do barril de óleo de raio 2 e altura 4: ")
(define V (VolumeOleo 2 4))
(display V)
```

Arquivo 05-funcoes.rkt

10. Execute o programa e explique o que faz

Print

The screenshot shows the DrRacket IDE with a file named '05-funcoes-a.rkt'. The editor contains a Racket script that defines two functions, 'ADICIONA' and 'soma', and displays their results. The output window on the right shows the welcome message and the execution results, including the function definitions and the results of the function calls.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem RRS
#lang racket ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;; DEFINIÇÃO DE FUNÇÕES-procedimentos usando lambda
;; (define NomeFuncao (lambda ( parametros ) (definicao) ))
;-----
;Execute o programa e explique o que faz:
#|
A Função "ADICIONA" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores.
|#
(define ADICIONA (lambda (a b) (+ a b) ))
;-----
A Função "soma" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (porém, com alguns textos demonstrativos extras).
|#
(define soma
  (lambda (x y)
    (begin
      (newline)
      (display "A soma de ")
      (display x)
      (display " e ")
      (display y)
      (display " = ")
      (+ x y)
    )
  ))
;-----
#|
A Função "produto" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (embora o nome da função indique uma operação

```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
40
70
A soma de 45 e 32 = 77
O produto de 21 e 15 = 36
A divisao = 427
A diferenca = 124
O quadrado = 61
>|
```

Código

#|

```

A Função "ADICIONA" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores.
|#
(define ADICIONA (lambda (a b) (+ a b) ) )
;;-----
#|
A Função "soma" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (porém, com alguns textos demonstrativos extras).
|#
(define soma
  (lambda (x y)
    (begin
      (newline)
      (display "A soma de ")
      (display x)
      (display " e ")
      (display y)
      (display " = ")
      (+ x y)
    )
  )
)

;;-----
#|
A Função "produto" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (embora o nome da função indique uma operação
diferente (Não sei se eu deveria alterar a operação para que ela faça o que diz de fato))
|#
(define produto
  (lambda (x y)
    (begin
      (newline)
      (display "O produto de ") (display x) (display " e ") (display y)
      (display " = ")
      (+ x y)
    )
  )
)

;;-----
(define divisao
  (lambda (x y)
    (begin
      (newline)
      (display "A divisao = ")
      (+ x y)
    )
  )
)

#|
A Função "divisao" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (embora o nome da função indique uma operação
diferente (Não sei se eu deveria alterar a operação para que ela faça o que diz de fato))
|#

```



```

#|
A Função "diferenca" recebe o valor de duas variáveis x e y,
imprime na tela a soma dos dois valores (embora o nome da função indique uma operação
diferente (Não sei se eu deveria alterar a operação para que ela faça o que diz de fato))
|#
(define diferenca
  (lambda (x y)
    (begin
      (newline)
      (display "A diferenca = ")
      (+ x y)
    )
  )
)
;;-----
#|
A Função "quadrado" recebe o valor de uma variável a,
imprime na tela o produto de a com a, ou seja, eleva a ao quadrado.
|#
(define quadrado
  (lambda ( a )
    (newline)
    (display "O quadrado = ")
    ( * a a)
  )
)
;;-----

;;----- Executando funcoes -----
#|
Nessa seção são feitos alguns testes de valores nas funções
|#

(ADICIONA 25 15)
(ADICIONA 30 40)

(soma 45 32)
(prodoto 21 15)
(divisao 420 7)
(diferenca 89 35)
(quadrado 09)

```

Explicação

São definidas algumas funções:

ADICIONA, soma, produto, divisão, diferença e quadrado.

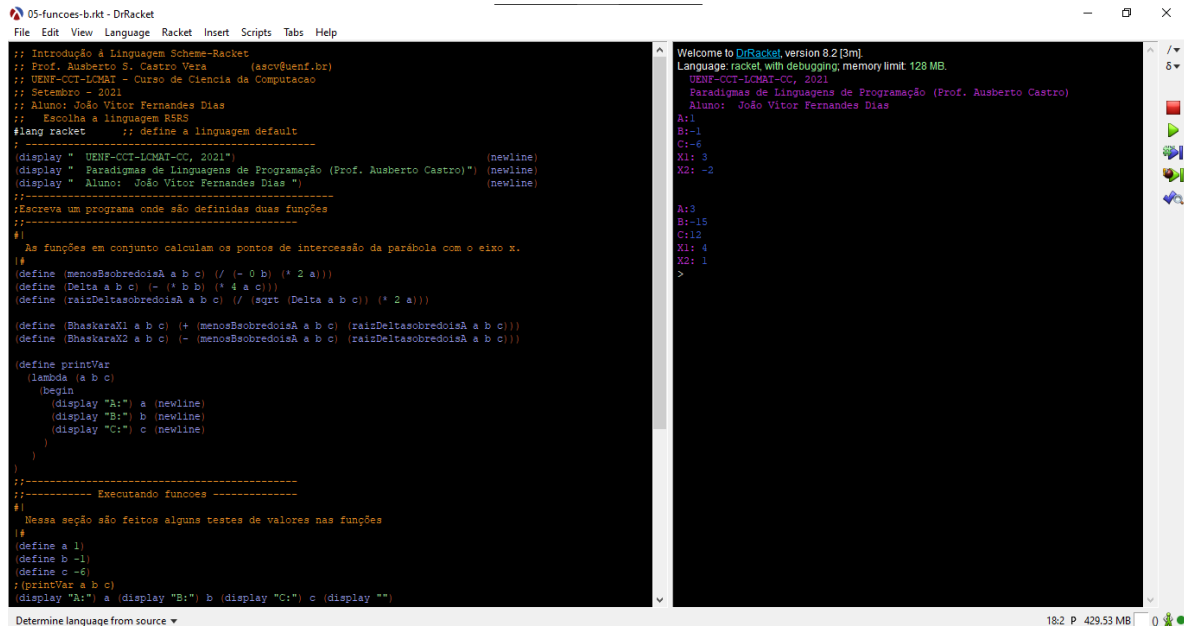
A função “quadrado” recebe o valor de uma variável e retorna o seu valor elevado ao quadrado (ou seja, multiplicado por ele mesmo).

Já as outras, apesar dos diferentes nomes, fazem a mesma coisa. Todas elas recebem o valor de duas variáveis e retornam a soma desses valores.

Já no final do código, são testadas as funções com diferentes valores.

11. Escreva um programa onde são definidas duas funções

Print



```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem R5RS
#lang racket ;; define a linguagem default

;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;-----

;Escreva um programa onde são definidas duas funções
;-----
#|
As funções em conjunto calculam os pontos de intercessão da parábola com o eixo x.
|#

(define (menosBsobredoisA a b c) (/ (- 0 b) (* 2 a)))
(define (Delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredoisA a b c) (/ (sqrt (Delta a b c)) (* 2 a)))

(define (BhaskaraX1 a b c) (+ (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))
(define (BhaskaraX2 a b c) (- (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))

(define printVar
  (lambda (a b c)
    (begin
      (display "A:" a (newline))
      (display "B:" b (newline))
      (display "C:" c (newline))
    )
  )
)

;----- Executando funcoes -----
#|
Nessa seção são feitos alguns testes de valores nas funções
|#
(define a 1)
(define b -1)
(define c -6)
(printVar a b c)
(display "A:") a (display "B:") b (display "C:") c (display ""))
```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; Escolha a linguagem R5RS
#lang racket ;; define a linguagem default

;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;-----

;Escreva um programa onde são definidas duas funções
;-----
#|
As funções em conjunto calculam os pontos de intercessão da parábola com o eixo x.
|#
(define (menosBsobredoisA a b c) (/ (- 0 b) (* 2 a)))
(define (Delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredoisA a b c) (/ (sqrt (Delta a b c)) (* 2 a)))

(define (BhaskaraX1 a b c) (+ (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))
(define (BhaskaraX2 a b c) (- (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))

(define printVar
```

```

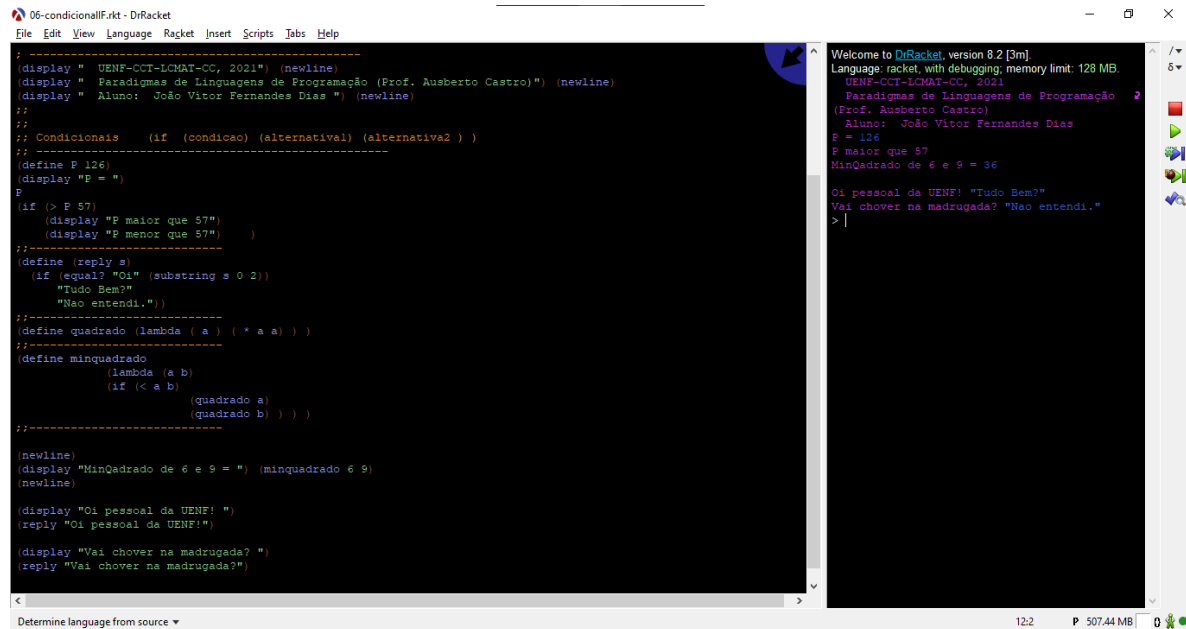
(lambda (a b c)
  (begin
    (display "A:") a (newline)
    (display "B:") b (newline)
    (display "C:") c (newline)
  )
)
)
;;-----
;;----- Executando funcoes -----
#|
Nessa seção são feitos alguns testes de valores nas funções
|#
(define a 1)
(define b -1)
(define c -6)
;(printVar a b c)
(display "A:") a (display "B:") b (display "C:") c (display "")
;(display "-b/2a: ")(menosBsobre doisA a b c)
;(display "b²-4ac: ")(Delta a b c)
;(display "raiz(delta)/2a: ")(raizDeltasobre doisA a b c)
(display "X1: ")(BhaskaraX1 a b c)
(display "X2: ")(BhaskaraX2 a b c)
(newline)(newline)
(define i 3)
(define j -15)
(define k 12)

;(printVar i j k)
(display "A:") i (display "B:") j (display "C:") k (display "")
;(display "-b/2a: ")(menosBsobre doisA a b c)
;(display "b²-4ac: ")(Delta a b c)
;(display "raiz(delta)/2a: ")(raizDeltasobre doisA a b c)
(display "X1: ")(BhaskaraX1 i j k)
(display "X2: ")(BhaskaraX2 i j k)

```

12. Executar e explicar o programa

Print



The screenshot shows the DrRacket IDE with a file named '06-condicionalIF.rkt'. The editor contains Racket code that defines several functions and performs calculations. The output window on the right shows the results of the program's execution.

```
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;;
;; Condiçionais (if (condicao) (alternativa1) (alternativa2) )
;;
(define P 126)
(display "P = ")
P
(if (> P 57)
  (display "P maior que 57")
  (display "P menor que 57") )
;;-----
(define (reply s)
  (if (equal? "Oi" (substring s 0 2))
      "Tudo Bem?"
      "Nao entendi."))
;;-----
(define quadrado (lambda (a) ( * a a ) ))
;;-----
(define minquadrado
  (lambda (a b)
    (if (< a b)
        (quadrado a)
        (quadrado b) ) ) )
;;-----
(newline)
(display "MinQadrado de 6 e 9 = ") (minquadrado 6 9)
(newline)
(display "Oi pessoal da UENF! ")
(reply "Oi pessoal da UENF!")
(display "Vai chover na madrugada? ")
(reply "Vai chover na madrugada?")
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação
(Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
P = 126
P maior que 57
MinQadrado de 6 e 9 = 36
Oi pessoal da UENF! "Tudo Bem?"
Vai chover na madrugada? "Nao entendi."
>|
```

Explicação

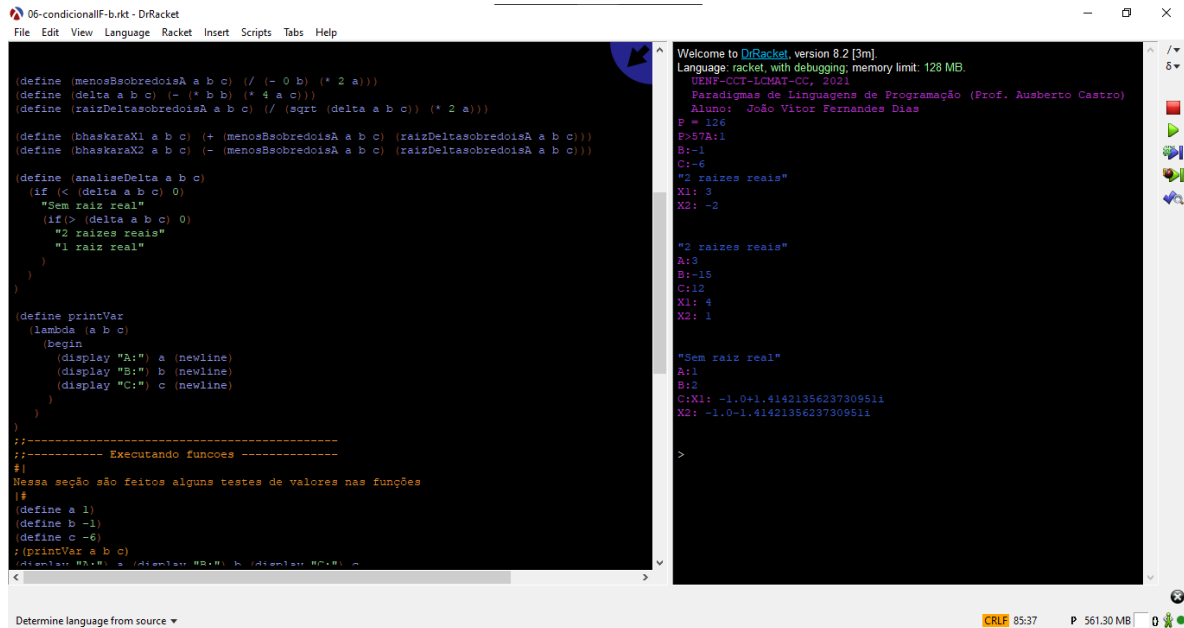
De uma forma geral, o programa exemplifica o uso da estrutura condicional if.

Mais especificamente ele inicia comparando o valor de uma variável e retornando textos diferentes de acordo com o seu valor. Mais abaixo define uma função chamada reply que responde “tudo bem” caso a substring indo do índice 0 ao 2 for igual a “Oi”, e caso não seja, responde “Não entendi”. Outras duas funções que trabalham em conjunto são definidas abaixo. Uma que retorna o menor valor entre dois números elevados ao quadrado e outra que calcula o valor de um número elevado ao quadrado.

Por fim, as funções são testadas.

13. Escreva um programa com dois condicionais

Print



```
(define (menosBsobredoisa a b c) (/ (- 0 b) (* 2 a)))
(define (delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredoisa a b c) (/ (sqrt (delta a b c)) (* 2 a)))

(define (bhaskaraX1 a b c) (+ (menosBsobredoisa a b c) (raizDeltasobredoisa a b c)))
(define (bhaskaraX2 a b c) (- (menosBsobredoisa a b c) (raizDeltasobredoisa a b c)))

(define (analiseDelta a b c)
  (if (< (delta a b c) 0)
      "Sem raiz real"
      (if (> (delta a b c) 0)
          "2 raizes reais"
          "1 raiz real")))

(define printVar
  (lambda (a b c)
    (begin
      (display "A:") a (newline)
      (display "B:") b (newline)
      (display "C:") c (newline))
    ))

;;----- Executando funcoes -----
#|
Nessa seção são feitos alguns testes de valores nas funções
|#
(define a 1)
(define b -1)
(define c -6)
(printVar a b c)
(display "A:") a (display "B:") b (display "C:") c)
```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vítor Fernandes Dias
P = 126
P>57A:1
B:-1
C:-6
"2 raizes reais"
X1: 3
X2: -2
"2 raizes reais"
A:1
B:-1
C:-6
X1: -1.0+1.4142135623730951i
X2: -1.0-1.4142135623730951i
>

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
;; Escolha a linguagem R5RS
;;
;; Ajuda: http://docs.racket-lang.org/guide/syntax-overview.html#(part._Conditionals_with_if_and_or_and_cond)
#lang racket ;; define a linguagem default
; -----Condicionais-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)

(define (menosBsobredoisa a b c) (/ (- 0 b) (* 2 a)))
(define (delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredoisa a b c) (/ (sqrt (delta a b c)) (* 2 a)))

(define (bhaskaraX1 a b c) (+ (menosBsobredoisa a b c) (raizDeltasobredoisa a b c)))
(define (bhaskaraX2 a b c) (- (menosBsobredoisa a b c) (raizDeltasobredoisa a b c)))

(define (analiseDelta a b c)
  (if (< (delta a b c) 0)
      "Sem raiz real"
      (if (> (delta a b c) 0)
          "2 raizes reais"
          "1 raiz real"))))

(define (execucao a b c) (newline)
  (display "A:") (display a) (display " B:") (display b) (display " C:") (display c)
  (display " "))
```

```
(display (analiseDelta a b c)) (newline)
(display "X1: ") (display (bhaskaraX1 a b c))
(display " X2: ") (display (bhaskaraX2 a b c)) (newline) )
;;----- Executando funcoes -----
; Nessa seção são feitos alguns testes de valores nas funções
(execucao 1 -1 -6)
(execucao 3 -15 12)
(execucao 1 2 3)
```

14. Escreva um programa para calcular a média de três notas e indique "Aprovado" se for maior ou igual a 6,0, e "Reprovado", caso contrário

Print

The screenshot shows the DrRacket IDE with a file named '06-condicionalIF-c.rkt - DrRacket'. The editor contains Scheme code for calculating the average of three numbers and determining if they are approved or not. The output window on the right shows the results of running the program, including the version of DrRacket (8.2 [3m]), the language (racket), and the output of the program: 'UENF-CCT-LCMAT-CC, 2021', 'Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)', 'Aluno: João Vítor Fernandes Dias', and the results of the 'aprovacao' function for three different sets of inputs: (5 5 5) resulting in 'aprovado', (5 6 7) resulting in 'aprovado', and (7 7 7) resulting in 'aprovado'.

```
06-condicionalIF-c.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias

#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)
;; Condicionais (if (condicao) (alternativa1) (alternativa2) )
; -----

(define (media v1 v2 v3) (/ (+ v1 v2 v3) 3))
(define (aprovacao v1 v2 v3) (if (>= (media v1 v2 v3) 6) "aprovado" "Reprovado"))
(aprovacao 5 5 5)
(aprovacao 5 6 7)
(aprovacao 7 7 7)
```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)
;; Condicionais (if (condicao) (alternativa1) (alternativa2) )
; -----

(define (media v1 v2 v3) (/ (+ v1 v2 v3) 3))
(define (aprovacao v1 v2 v3) (if (>= (media v1 v2 v3) 6) "aprovado" "Reprovado"))
(aprovacao 5 5 5)
(aprovacao 5 6 7)
(aprovacao 7 7 7)
```

15. Executar e explicar o programa

- DÚVIDA

Print

```

07-formulas.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-OCT-LCMAT - Curso de Ciência da Computação
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;;

#lang racket      ;; define a linguagem default
;
;-----
(display " UENF-OCT-LCMAT-CC, 2021")
(newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)")
(newline)
(display " Aluno: João Vitor Fernandes Dias ")
(newline)
;;
;;Algumas formulas conhecidas
;-----
;; Formula de Pitagoras
(define (pitagoras a b)
  (+ (* a a) (* b b)))
;-----
;; Aproximacao do numero Pi
(define (aprox-pi N)
  (if (zero? N)
      4
      (+ (* 4 (/ (expt -1 N) (+ (* 2 N) 1)))
         (aprox-pi (- N 1))))
  )
;-----
;; Factorial de um numero inteiro N
(define (factorial N)
  (if (zero? N)
      1
      (* N (factorial (- N 1)))))

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-OCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Pitagoras de 3 e 4 = 25

Aproximando Pi com 4 = 3.107
                      315

Aproximando Pi com 10 = 3.3380087
                      14549535

Factorial de 5 = 120

Fazer outro teste sobre Pitagoras e outro sobre Factorial
>

```

Explicação

São definidas três funções:

A primeira retorna a soma dos quadrados de dois números ($a^2 + b^2$)

A segunda faz uma recursividade de tamanho N para encontrar um valor próximo do PI (Não entendi como funciona matematicamente - DÚVIDA)

A terceira calcula recursivamente o valor fatorial

E por fim são testadas essas funções com alguns valores de teste.

16. Escreva um programa que calcule o fatorial de um número de uma forma diferente da apresentada.

Print

```

07-formulas.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

(if (zero? N)
    4
    (+ (* 4 (/ (expt -1 N) (+ (* 2 N) 1))) (aprox-pi (- N 1)))
      ; 4*(-1^N) / (2^N+1) + aprox-pi(N-1)
    )
;-----
;; Factorial de um numero inteiro N
(define (factorial N)
  (if (zero? N)
      1
      (* N (factorial (- N 1)))))
;-----
;; Factorial de um numero inteiro N
(define (factorialR cont N)
  (if (<= cont N)
      (* cont (factorialR (+ cont 1) N))
      1
  )
)
(define (factorial N)
  (factorialR 1 N)
)

(display "Pitagoras de 3 e 4 = ") (pitagoras 3 4) (newline)
(display "Aproximando Pi com 4 = ") (aprox-pi 4) (newline)
(display "Aproximando Pi com 10 = ") (aprox-pi 10) (newline)
(display "Factorial de 5 = ") (factorial 5) (newline)
(display "Fazer outro teste sobre Pitagoras e outro sobre Factorial") (newline)
(display "Pitagoras de 6 e 8 = ") (pitagoras 6 8) (newline)
(display "Factorial de 6 = ") (factorial 6) (newline)
(newline) (newline)

(display "Factorial JV 5 = ") (factorial 5) (newline)
(display "Factorial JV 6 = ") (factorial 6) (newline)

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-OCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Pitagoras de 3 e 4 = 25

Aproximando Pi com 4 = 3.107
                      315

Aproximando Pi com 10 = 3.3380087
                      14549535

Factorial de 5 = 120

Fazer outro teste sobre Pitagoras e outro sobre Factorial
Pitagoras de 6 e 8 = 100
Factorial de 6 = 720

Factorial JV 5 = 120
Factorial JV 6 = 720

>

```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera    (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
;;
#lang racket    ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)(newline)
;;Algumas formulas conhecidas

;;-----
;; Formula de Pitagoras
(define (pitagoras a b) (+ (* a a) (* b b)) )

;;-----
;; Aproximacao do numero Pi
(define (aprox-pi N)
  (if (zero? N)
      4
      (+ (* 4 (/ (expt -1 N) (+ (* 2 N) 1))) (aprox-pi (- N 1)))
        ; 4*(-1^N)/(2*N+1) + aprox-pi(N-1)
  )
)
;;-----
;; factorial de um numero inteiro N
(define (factorial N)
  (if (zero? N)
      1
      (* N (factorial (- N 1)))
  )
)
;;-----
;; fatorial de um numero inteiro N
(define (fatorialR cont N)
  (if (<= cont N)
      (* cont (fatorialR (+ cont 1) N))
      1
  )
)
(define (fatorial N)
```



```
(fatorialR 1 N)
)
```

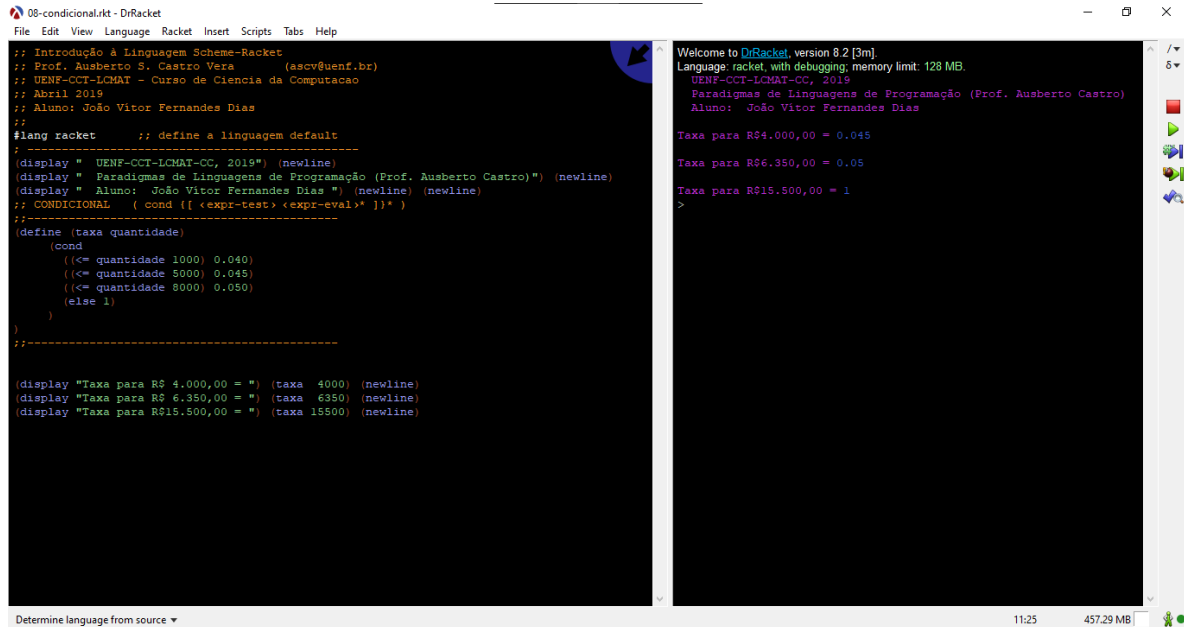
```
(display "Pitagoras de 3 e 4 = ") (pitagoras 3 4) (newline)
(display "Aproximando Pi com 4 = ") (aprox-pi 4) (newline)
(display "Aproximando Pi com 10 = ") (aprox-pi 10) (newline)
(display "Factorial de 5 = ") (factorial 5) (newline)
(display "Fazer outro teste sobre Pitagoras e outro sobre Fatorial") (newline)
(display "Pitagoras de 6 e 8 = ") (pitagoras 6 8)
(display "Factorial de 6 = ") (factorial 6) (newline)
(newline)(newline)
```

```
(display "Fatorial JV 5 = ") (factorial 5) (newline)
(display "Fatorial JV 6 = ") (factorial 6) (newline)
```

Arquivo 08-condicional.rtk

17. Execute o programa e indique o que faz

Print



The screenshot shows the DrRacket IDE with a file named '08-condicional.rtk'. The editor displays a Racket program that defines a function 'taxa' using a 'cond' expression. The program also includes several 'display' statements to show the results of the function for different input values. The output window on the right shows the execution results, including a welcome message and the calculated tax values for three different quantities.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciência da Computação
;; Abril 2019
;; Aluno: João Vitor Fernandes Dias
;;
;;#lang racket ;; define a linguagem default
;;
;;-----
(display " UENF-CCT-LCMAT-CC, 2019") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; CONDICIONAL ( cond [[ <expr-test> <expr-eval> ]]*)
;;-----
(define (taxa quantidade)
  (cond
    ((<= quantidade 1000) 0.040)
    ((<= quantidade 5000) 0.045)
    ((<= quantidade 8000) 0.050)
    (else 1)
  )
)
;;-----

(display "Taxa para R$ 4.000,00 = ") (taxa 4000) (newline)
(display "Taxa para R$ 6.350,00 = ") (taxa 6350) (newline)
(display "Taxa para R$15.500,00 = ") (taxa 15500) (newline)
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2019
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
Taxa para R$4.000,00 = 0.040
Taxa para R$6.350,00 = 0.05
Taxa para R$15.500,00 = 1
>
```

Explicação

O programa ilustra o uso da estrutura condicional-cond.

Ele funciona de maneira similar à estrutura switch...case da linguagem C, onde, baseado em uma variável específica (nesse caso a variável é “quantidade”, da função “taxa”) estabelece-se diversos casos possíveis, cada um com uma execução de funções ou retorno de valores específicos. Outra forma de explicar seria com a ideia de encadeamento de if's, onde pode-se colocar várias condicionais inseridas uma na outra a fim de haver várias alternativas possíveis, porém com a estrutura cond, isso se torna mais denso sintaticamente.

18. Escreva um programa condicional com pelo menos 5 opções

Print



The screenshot shows the DrRacket IDE with a Racket program on the left and its output on the right. The program defines a function `ordem` that takes a number `x` and returns a string representing its order of magnitude. The output shows the function being called with various values and returning the corresponding strings.

```
;; Abril 2019
;; Aluno: João Vitor Fernandes Dias
;;
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2019") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; CONDICIONAL ( cond {[ <expr-test> <expr-eval>* ]}* )
;;-----

#|
Escreva um programa condicional com pelo menos 5 opções
|#
(define (E10 x) (expt 10 x))

(define (ordem x) |
  (cond
    ((< x 0) "Negativo")
    ((and (>= x 0) (< x (E10 1))) "Unidade")
    ((< x (E10 2)) "Dezena")
    ((< x (E10 3)) "Centena")
    ((< x (E10 4)) "Unidade de milhar")
    ((< x (E10 5)) "Dezena de milhar")
    (else "Maior que Dezena de milhar")
  )
)
;;-----

(display "Ordem para o valor -2 = ") (ordem -2) (newline)
(display "Ordem para o valor 2 = ") (ordem 2) (newline)
(display "Ordem para o valor 20 = ") (ordem 20) (newline)
(display "Ordem para o valor 200 = ") (ordem 200) (newline)
(display "Ordem para o valor 2000 = ") (ordem 2000) (newline)
(display "Ordem para o valor 20000 = ") (ordem 20000) (newline)
(display "Ordem para o valor 200000 = ") (ordem 200000) (newline)
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-cc, 2019
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Ordem para o valor -2 = "Negativo"
Ordem para o valor 2 = "Unidade"
Ordem para o valor 20 = "Dezena"
Ordem para o valor 200 = "Centena"
Ordem para o valor 2000 = "Unidade de milhar"
Ordem para o valor 20000 = "Dezena de milhar"
Ordem para o valor 200000 = "Maior que Dezena de milhar"
>
```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Abril 2019
;; Aluno: João Vítor Fernandes Dias
;;
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2019") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
;; CONDICIONAL ( cond {[ <expr-test> <expr-eval>* ]}* )
;;-----

#|
Escreva um programa condicional com pelo menos 5 opções
|#
(define (E10 x) (expt 10 x))

(define (ordem x)
  (cond
    ((< x 0) "Negativo")
    ((and (>= x 0) (< x (E10 1))) "Unidade")
    ((< x (E10 2)) "Dezena")
    ((< x (E10 3)) "Centena")
    ((< x (E10 4)) "Unidade de milhar")
    ((< x (E10 5)) "Dezena de milhar")
    (else "Maior que Dezena de milhar")
  )
)
;;-----
```

```
(display "Ordem para o valor -2 = ") (ordem -2) (newline)
(display "Ordem para o valor 2 = ") (ordem 2) (newline)
(display "Ordem para o valor 20 = ") (ordem 20) (newline)
(display "Ordem para o valor 200 = ") (ordem 200) (newline)
(display "Ordem para o valor 2000 = ") (ordem 2000) (newline)
(display "Ordem para o valor 20000 = ") (ordem 20000) (newline)
(display "Ordem para o valor 200000 = ") (ordem 200000) (newline)
```

19. Escreva um programa bhaskara.rkt que calcule as raízes de uma equação $25x^2 - 55x + 10 = 0$, utilizando a fórmula de Bhaskara. Sugestão: Primeiro faça o algoritmo completo

Print

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Abril 2019
;; Aluno: João Vitor Fernandes Dias
;;
#lang racket ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2019") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; CONDICIONAL ( cond {[ <expr-test> <expr-eval>* ]}* )
;-----

#|
Escreva um programa bhaskara.rkt que calcule as raízes de uma equação
25x² - 55x + 10 = 0, utilizando a fórmula de Bhaskara.
Sugestão: Primeiro faça o algoritmo completo
|#

(define (menosBsobredoisA a b c) (/ (- 0 b) (* 2 a)))
(define (delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredoisA a b c) (/ (sqrt (delta a b c)) (* 2 a)))

(define (bhaskaraX1 a b c) (+ (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))
(define (bhaskaraX2 a b c) (- (menosBsobredoisA a b c) (raizDeltasobredoisA a b c)))

(define (analiseDelta a b c)
  (cond
    ((< (delta a b c) 0) "Sem raiz real")
    ((> (delta a b c) 0) "2 raizes reais")
    (else "1 raiz real")
  )
)

#|
```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Abril 2019
;; Aluno: João Vítor Fernandes Dias
;;
#lang racket ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2019") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
;; CONDICIONAL ( cond {[ <expr-test> <expr-eval>* ]}* )
;-----

#|
Escreva um programa bhaskara.rkt que calcule as raízes de uma equação
25x² - 55x + 10 = 0, utilizando a fórmula de Bhaskara.
Sugestão: Primeiro faça o algoritmo completo
|#
```

```

(define (menosBsobredeoisA a b c) (/ (- 0 b) (* 2 a)))
(define (delta a b c) (- (* b b) (* 4 a c)))
(define (raizDeltasobredeoisA a b c) (/ (sqrt (delta a b c)) (* 2 a)))

(define (bhaskaraX1 a b c) (+ (menosBsobredeoisA a b c) (raizDeltasobredeoisA a b c)))
(define (bhaskaraX2 a b c) (- (menosBsobredeoisA a b c) (raizDeltasobredeoisA a b c)))

(define (analiseDelta a b c)
  (cond
    ((< (delta a b c) 0) "Sem raiz real")
    ((> (delta a b c) 0) "2 raizes reais")
    (else "1 raiz real")
  )
)

#|
(define (analiseDelta a b c)
  (if (< (delta a b c) 0)
    "Sem raiz real"
    (if (> (delta a b c) 0)
      "2 raizes reais"
      "1 raiz real"
    )
  )
)
|#
;;----- Executando funcoes -----
#|
Nessa seção são feitos alguns testes de valores nas funções
#
(define a 25)
(define b -55)
(define c 10)
;(printVar a b c)
(display "A:") a (display "B:") b (display "C:") c
;(display "-b/2a: ")(menosBsobredeoisA a b c)
;(display "b²-4ac: ")(Delta a b c)
;(display "raiz(delta)/2a: ")(raizDeltasobredeoisA a b c)
(analiseDelta a b c)
(display "X1: ")(bhaskaraX1 a b c)
(display "X2: ")(bhaskaraX2 a b c)
(newline)(newline)

```

20. Execute o programa

15h21

Print

```

09-pares.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;;

#lang racket      ;; define a linguagem default
;
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro), 2021") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)(newline)

;; -----
;; PARES CONSTANTES
;; -----
(define x (cons 1 2))
(define y (cons 'a 3))
(define z (cons x y)) ; ( (1 2) ('a 3) )

(define a (cons 1 2)) ; (1 2)
(define a2 (cons a 3)) ; ((1 2) 3)
(define a3 (cons a2 4)) ; (((1 2) 3) 4)
(define a4 (cons a3 5)) ; ((((1 2) 3) 4) 5)

(define b (cons 2 1)) ; (2 1)
(define b2 (cons 3 b)) ; (3 (2 1))
(define b3 (cons 4 b2)) ; (4 (3 (2 1)))
(define b4 (cons 5 b3)) ; (5 (4 (3 (2 1))))

;; -----
;; Primeiro e resto
;; -----
(display "O par x = ") x (newline)
(display "O par y = ") y (newline)
(display "O par z = (x y) = ") z (newline)(newline)
(display "O par a4 = ") a4 (newline)
(display "O par b4 = ") b4 (newline)

;; -----
;; Primeiro e resto
;; -----
(display "O primeiro elemento do par x = ") (car x) (newline) ; primeiro
(display "O segundo elemento do par x = ") (cdr x) (newline) ; resto

```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro), 2021
Aluno: João Vitor Fernandes Dias

O par x = '(1 . 2)
O par y = '(a . 3)
O par z = (x y) = '(((1 . 2) a) . 3)
O par a4 = '((((1 . 2) . 3) . 4) . 5)
O par b4 = '(5 4 3 2 . 1)
O primeiro elemento do par x = 1
O segundo elemento do par x = 2
>

21. Escreva um NOVO programa para construir dois pares e indicar em cada um deles o primeiro e o segundo elemento

15h22 – 15h32

Print

```

09-pares-b.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;;

#lang racket      ;; define a linguagem default
;
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro), 2021") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)(newline)

;; -----
;; PARES CONSTANTES
;; -----
(define x (cons 'x1 'x2))
(define y (cons 'y1 'y2))

;; -----
;; Primeiro e resto
;; -----
(define (primeiroElemento X) (car X))
(define (segundoElemento X) (cdr X))

;; -----
;; Primeiro e resto
;; -----
(display "O primeiro elemento do par x = ") (primeiroElemento x) (newline) ; 1x
(display "O segundo elemento do par x = ") (segundoElemento x) (newline) ; 2x
(display "O primeiro elemento do par y = ") (primeiroElemento y) (newline) ; 1y
(display "O segundo elemento do par y = ") (segundoElemento y) (newline) ; 2y

```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro), 2021
Aluno: João Vitor Fernandes Dias

O primeiro elemento do par x = 'x1
O segundo elemento do par x = 'x2
O primeiro elemento do par y = 'y1
O segundo elemento do par y = 'y2
>

Código

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias

```



```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
#lang racket ;; define a linguagem default

;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (new
; O procedimento CONS constrói listas e tem DOIS parametros
(cons 'a '(x y w u))
(define lista1 (cons 'a '(x y w u)))
(define lista2a (cons 'x (cons 'a (cons 'm '(3 7)))))
(define lista2b (cons 'x (cons 'a (cons 'm (cons 3 7)))))
(display "Lista2a:") lista2a (newline)
(display "Lista2b:") lista2b (newline)
;-----
(display "O ponto, antes do ultimo elemento, numa lista, significa LISTA IMPROPRIA...!") (n
(cons 'a 'b)
(cdr '(a . b)) ; lista UM elemento
(cons 'a '(b . c)) (newline) ; lista 2 elementos
;-----
(list 'a 'b 'c) ; lista 3 elementos
(list 'x) ; lista unitaria
(list) ; lista vazia

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

'(a x y w u)
Lista2a: '(x a m 3 7)
Lista2b: '(x a m 3 . 7)

O ponto, antes do ultimo elemento, numa lista, significa LISTA
IMPROPRIA...!
'(a . b)
'b
'(a b . c)
'(a b c)
'(x)
'()
>
```

23. Escreva um NOVO programa para construir uma lista e determinar seu primeiro e último elemento, seu comprimento, e uma nova lista com dois elementos a mais que a anterior. Incluir os códigos fonte 16h33 – 17h16

Print

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
#lang racket ;; define a linguagem default

;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (new
;-----Descrição do problema-----
;Escreva um NOVO programa para
;construir uma lista
;e determinar seu primeiro e último elemento,
;seu comprimento,
;e uma nova lista com dois elementos a mais que a anterior.
;Incluir os códigos fonte
;-----Definição da lista-----
(define lista1 (list 0 1 2 3 4 5 6 7))
;-----Funções-----
(define primeiroElemento X) (car X)
(define (ultimoElemento X) (car (reverse X)))
(define (tamanhoLista X) (length X))
(define (insere2 X) (append X (list (ultimoElemento X) (ultimoElemento X))))
;-----Testes-----
(newline) (display "Primeiro elemento: ") (primeiroElemento lista1)
(newline) (display "Ultimo elemento: ") (ultimoElemento lista1)
(newline) (display "TamanhoLista: ") (tamanhoLista lista1)
(newline) (display "Insere2: ") (insere2 lista1)

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Primeiro elemento: 0
Ultimo elemento: 7
TamanhoLista: 8
Insere2: '(0 1 2 3 4 5 6 7 7 7)
>
```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
```

```
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
; -----Descrição do problema-----
;Escreva um NOVO programa para
;construir uma lista
;e determinar seu primeiro e último elemento,
;seu comprimento,
;e uma nova lista com dois elementos a mais que a anterior.
;Incluir os códigos fonte
; -----Definição da lista-----
(define Lista1 (list 0 1 2 3 4 5 6 7))
; -----Funções-----
(define (primeiroElemento X) (car X))
(define (ultimoElemento X) (car (reverse X)))
(define ( tamanhoLista X) (length X))
(define ( insere2 X) (append X (list (ultimoElemento X) (ultimoElemento X))))
; -----Testes-----
(newline) (display "Primeiro elemento: ") (primeiroElemento Lista1)
(newline) (display "Ultimo elemento: ") ( ultimoElemento Lista1)
(newline) (display "TamanhoLista: ") ( tamanhoLista Lista1)
(newline) (display "Insere2: ") ( insere2 Lista1)
```

24. Utilizando uma ÚNICA linha de comandos, escreva um NOVO programa Racket para construir a lista (4 7 2 9 8 7 1 6 2 3 4) a partir das listas A= (1 2 3 4) e B= (5 6 7 8 9) 17h16 – 17h51

Print

The screenshot shows the DrRacket IDE interface. The editor window on the left contains the Racket code. The output window on the right shows the program's execution, displaying the student's name and the list '(4 7 2 9 8 7 1 6 2 3 4)'.

Código

#|Introdução à Linguagem Scheme-Racket Prof. Ausberto S. Castro Vera
(ascv@uenf.br)UENF-CCT-LCMAT - Curso de Ciencia da ComputacaoSetembro -
2021Aluno: João Vítor Fernandes Dias|# #lang racket (display "UENF-CCT-LCMAT-CC,
2021")(newline)(display "Paradigmas de Linguagens de Programação (Prof. Ausberto

Castro)))(newline)(display "Aluno: João Vítor Fernandes Dias ")(newline) (newline)#|
 Utilizando uma ÚNICA linha de comandos, escreva um NOVO programa Racket para
 construir a lista (4 7 2 9 8 7 1 6 2 3 4) a partir das listas A= (1 2 3 4) e B= (5 6 7 8 9) |#(let
 ((A (list 1 2 3 4)) (B (list 5 6 7 8 9)))(list(car (reverse A))(car (cdr (cdr B)))(car (cdr A))(car
 (reverse B))(car (cdr (reverse B)))(car (cdr (cdr B)))(car A)(car (cdr B))(car (cdr A))(car (cdr
 (reverse A)))(car (reverse A))))

A função especificamente:

(let ((A (list 1 2 3 4)) (B (list 5 6 7 8 9)))(list(car (reverse A))(car (cdr (cdr B)))(car (cdr A))(car
 (reverse B))(car (cdr (reverse B)))(car (cdr (cdr B)))(car A)(car (cdr B))(car (cdr A))(car (cdr
 (reverse A)))(car (reverse A))))

Arquivo 12-lambda.rkt e 13-lambda.rkt

19h34 – 20h51

25. Execute os programas e indique o que faz cada um deles 19h34 – 20h09

25.1. Arquivo 12-lambda.rkt 19h34 – 19h43

Print

The screenshot shows the DrRacket IDE with a file named '12-lambda.rkt'. The editor contains Racket code that defines several functions and performs calculations. The output window on the right shows the results of these operations.

```

(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)

;; -----
;; Expressões LAMBDA
;; Expressões LAMBDA são utilizadas para criar novos procedimentos
;; (lambda (var ...) expr1 expr2 ....)
;; ----- função Duas-vezes -----
(define duasvezes
  (lambda (f x) ;; <----- 2 parametros f x
    (f x x) ;; <----- definição da função
  ))

;; ----- função com5 -----
(define com5
  (lambda (operador x)
    (operador x 5)
  ))

;; ----- Polinômio P(x) = X^2 + 3X - 7 -----
(define polinomio
  (lambda (x) ;; <----- um parametro x
    (- (+ (* x x) (* 3 x)) 7)
  ))

;; ----- executando ... -----
(display "3+3 = ") (duasvezes + 3)
(display "3*3 = ") (duasvezes * 3)
(display "3-3 = ") (duasvezes - 3) (newline)

(display "8+5 = ") (com5 + 8)
(display "8*5 = ") (com5 * 8)
(display "8-5 = ") (com5 - 8) (newline)

(display "P(x) = X^2 + 3X - 7 então P(5) = ") (polinomio 5)
(display "P(x) = X^2 + 3X - 7 então P(0) = ") (polinomio 0)
(display "P(x) = X^2 + 3X - 7 então P(2) = ") (polinomio 2)
  
```

The output window displays the following results:

```

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UNFP-CCT-ICMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vítor Fernandes Dias

3+3 = 6
3*3 = 9
3-3 = 0

8+5 = 13
8*5 = 40
8-5 = 3

P(x) = X^2 + 3X - 7 então P(5) = 33
P(x) = X^2 + 3X - 7 então P(0) = -7
P(x) = X^2 + 3X - 7 então P(2) = 3
>|
  
```

Explicação

Este programa ilustra o uso do lambda (embora eu não tenha entendido ainda qual a necessidade do uso dele ao invés do uso padrão do define).

É criada uma função chamada “duasvezes” que faz uma devida operação de um valor com ele mesmo. Tanto a operação, quanto o valor são enviados como parâmetros.

Em seguida é criada uma função “com5” que faz uma devida operação de um valor com o 5, sendo sempre o 5 o segundo termo da operação. Tanto a operação, quanto o valor são enviados como parâmetros.

A próxima função definida é a função “polinomio” que retorna o valor da operação “x²+3x-7”, sendo x uma variável que foi enviada como parâmetro.

Por fim, são testadas as funções com diversos valores e operações diferentes.

Print

The screenshot shows the DrRacket IDE with a Racket script on the left and its execution output on the right.

Script Content (Left Panel):

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciência da Computação
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;;
#lang racket      ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)
;-----
;; Expressões LAMBDA e Recursão
;-----
;; Expressões LAMBDA são utilizadas para criar novos procedimentos
;; (lambda (var ...) expr1 expr2 ....)
;; tamanho de uma lista -----
(define comprimento
  (lambda (lista)
    (if (null? lista)
        0
        (+ (comprimento (cdr lista)) 1)
    )
  )
)
;----- remove um elemento de uma lista -----
(define remove
  (lambda (x lista)
    (cond
      ((null? lista) '())
      ((eq? (car lista) x) (remove x (cdr lista)))
      (else (cons (car lista) (remove x (cdr lista)))))
  )
)
;-----
(define inverso
  (lambda (n)
    (if (and (number? n) (not (= n 0)))
        (/ 1 n)
    )
  )
)

```

Execution Output (Right Panel):

```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-cc, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

lista: '(1 2 3 4 5 6 7)
tamanho da lista: 7
Removendo um elemento (o 5): '(1 2 3 4 6 7)
Inverso de 12 = 1/12
> |

```

Explicação

A primeira função calcula o comprimento da lista (assim como o length) de forma recursiva. Enquanto a lista não estiver vazia, somará 1 e chamará novamente a função, enviando todos os elementos, menos o primeiro, como parâmetro. Quando estiver vazia, retornará 0.

A segunda função faz a busca na lista e quando o encontrar, removerá (só que de forma mais complexa e recursiva). Ele recebe como parâmetro o valor que está sendo buscado e a lista, então entra na condicional: caso a lista estiver vazia, retornará uma lista vazia. Caso o primeiro elemento da lista for equivalente ao valor que está sendo buscado, chama-se novamente a função enviando como parâmetro toda a lista após o primeiro elemento (é dessa forma que o elemento é removido). Caso não seja nenhum dos casos acima, ele adiciona o primeiro elemento da lista ao retorno da chamada recursiva da função “remove” que envia o valor a ser removido e todo o resto da lista.

A terceira função, chamada “inverso” confere se o valor n recebido é um número e se ele não é igual a zero, se ambas as condições forem verdadeiras, ele retornará o valor referente ao inverso no número n, ou seja, $1/n$.

Abaixo é criada a lista e testadas as funções descritas acima com valores de teste.

26. Crie um procedimento para realizar o cálculo de uma prestação em atraso, utilizando a fórmula $\text{Prest} = \text{valor} + (\text{valor} * (\text{taxa}/100) * \text{tempo})$. Dar exemplos. 20h09 – 20h23]

Print

The screenshot shows the DrRacket IDE with a Racket program on the left and its output on the right. The program defines a function `prestacaoAtrasada` that takes a value and a tax rate, and then displays a table of results for various values and tax rates. The output shows the results of these calculations, including the value, tax rate, and the resulting amount.

```

(define prestacaoAtrasada
  (lambda (valor taxa tempo)
    (+ valor (* valor tempo (/ taxa 100)))))

;----- Executando -----
(display "Valor 1000, Taxa 5, Tempo 4: ") (prestacaoAtrasada 1000 5 4)
(display "Valor 1000, Taxa 5, Tempo 8: ") (prestacaoAtrasada 1000 5 8)
(display "Valor 1000, Taxa 5, Tempo 12: ") (prestacaoAtrasada 1000 5 12)
(display "Valor 1000, Taxa 10, Tempo 4: ") (prestacaoAtrasada 1000 10 4)
(display "Valor 1000, Taxa 10, Tempo 8: ") (prestacaoAtrasada 1000 10 8)
(display "Valor 1000, Taxa 10, Tempo 12: ") (prestacaoAtrasada 1000 10 12)
(display "Valor 1000, Taxa 15, Tempo 4: ") (prestacaoAtrasada 1000 15 4)
(display "Valor 1000, Taxa 15, Tempo 8: ") (prestacaoAtrasada 1000 15 8)
(display "Valor 1000, Taxa 15, Tempo 12: ") (prestacaoAtrasada 1000 15 12)
(newline)
(display "Valor 10000, Taxa 5, Tempo 4: ") (prestacaoAtrasada 10000 5 4)
(display "Valor 10000, Taxa 5, Tempo 8: ") (prestacaoAtrasada 10000 5 8)
(display "Valor 10000, Taxa 5, Tempo 12: ") (prestacaoAtrasada 10000 5 12)
(display "Valor 10000, Taxa 10, Tempo 4: ") (prestacaoAtrasada 10000 10 4)
(display "Valor 10000, Taxa 10, Tempo 8: ") (prestacaoAtrasada 10000 10 8)
(display "Valor 10000, Taxa 10, Tempo 12: ") (prestacaoAtrasada 10000 10 12)
(display "Valor 10000, Taxa 15, Tempo 4: ") (prestacaoAtrasada 10000 15 4)
(display "Valor 10000, Taxa 15, Tempo 8: ") (prestacaoAtrasada 10000 15 8)
(display "Valor 10000, Taxa 15, Tempo 12: ") (prestacaoAtrasada 10000 15 12)
(newline)
(display "Valor 100000, Taxa 5, Tempo 4: ") (prestacaoAtrasada 100000 5 4)
(display "Valor 100000, Taxa 5, Tempo 8: ") (prestacaoAtrasada 100000 5 8)
(display "Valor 100000, Taxa 5, Tempo 12: ") (prestacaoAtrasada 100000 5 12)
(display "Valor 100000, Taxa 10, Tempo 4: ") (prestacaoAtrasada 100000 10 4)
(display "Valor 100000, Taxa 10, Tempo 8: ") (prestacaoAtrasada 100000 10 8)
(display "Valor 100000, Taxa 10, Tempo 12: ") (prestacaoAtrasada 100000 10 12)
(display "Valor 100000, Taxa 15, Tempo 4: ") (prestacaoAtrasada 100000 15 4)
(newline)

```

Output:

```

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
Valor 1000, Taxa 5, Tempo 4: 1200
Valor 1000, Taxa 5, Tempo 8: 1400
Valor 1000, Taxa 5, Tempo 12: 1600
Valor 1000, Taxa 10, Tempo 4: 1400
Valor 1000, Taxa 10, Tempo 8: 1800
Valor 1000, Taxa 10, Tempo 12: 2200
Valor 1000, Taxa 15, Tempo 4: 1600
Valor 1000, Taxa 15, Tempo 8: 2200
Valor 1000, Taxa 15, Tempo 12: 2800
Valor 10000, Taxa 5, Tempo 4: 12000
Valor 10000, Taxa 5, Tempo 8: 14000
Valor 10000, Taxa 5, Tempo 12: 16000
Valor 10000, Taxa 10, Tempo 4: 14000
Valor 10000, Taxa 10, Tempo 8: 18000
Valor 10000, Taxa 10, Tempo 12: 22000
Valor 10000, Taxa 15, Tempo 4: 16000
Valor 10000, Taxa 15, Tempo 8: 22000
Valor 10000, Taxa 15, Tempo 12: 28000
Valor 100000, Taxa 5, Tempo 4: 120000
Valor 100000, Taxa 5, Tempo 8: 140000
Valor 100000, Taxa 5, Tempo 12: 160000
Valor 100000, Taxa 10, Tempo 4: 140000
Valor 100000, Taxa 10, Tempo 8: 180000
Valor 100000, Taxa 10, Tempo 12: 220000

```

27. O que faz o seguinte procedimento abcd:

20h23 – 20h51

Procedimento

```

(define abcd
  (lambda (n)
    (let f ((i 2))
      (cond
        ((>= i n) '())
        ((integer? (/ n i))
         (cons i (f (+ i 1))))
        (else (f (+ i 1)))))))

```

Print

The screenshot shows the Visual Studio Code editor with a Racket program. The program is a lambda function named `abcd` that takes a number `n` and returns a list of its divisors. The code is annotated with comments in Portuguese explaining the logic of the function.

```

29 (define abcd
30   (lambda (n) .....;recebe um valor n
31     (let f ((i 2)) .....;dentro desse escopo torna o valor i igual a 2
32       .....;Porém não sei qual a função do f nesse contexto
33       .....;
34       .....;Após rodar o código, entendi que o é uma função que está
35       .....;sendo definida e está sendo utilizada na recursão,
36       .....;utilizando o i como contador que vai de 2 a n, porém,
37       .....;ainda não entendi de que forma este f está sendo definido
38       .....;
39       (cond .....;ocorre uma condicional
40         ((>= i n) '()) .....;caso i (que equivale a 2) seja maior ou igual ao valor
41         ((integer? (/ n i)) .....;recebido, retorna uma lista vazia
42         ((cons i (f (+ i 1))) .....;caso n/i seja inteiro,
43         .....;cria um par contendo i e o resultado de uma função f
44         .....;que envia como parâmetro o valor i+1
45         (else (f (+ i 1))) .....;caso não seja nenhuma das situações acima, chama a função f
46         .....;que envia como parâmetro o valor i+1
47       )
48     )
49 )

```

Explicação

Após alguns testes que são descritos de forma menos formal abaixo, pode-se compreender que este código retorna uma lista contendo todos os múltiplos de um determinado valor n , excluindo os valores 1 e o próprio valor n .

Código

```
(define abcd
  (lambda (n)           ;recebe um valor n
    (let f ((i 2))      ;dentro desse escopo torna o valor i igual a 2
      ;Porém não sei qual a função do f nesse contexto
      ;(
      ; Após rodar o código, entendi que o é uma função que está
      ; sendo definida e está sendo utilizada na recursão,
      ; utilizando o i como contador que vai de 2 a n, porém,
      ; ainda não entendi de que forma este f está sendo definido
      ;)
      (cond              ;ocorre uma condicional
        ((>= i n) '())   ;caso i (que equivale a 2) seja maior ou igual ao valor
        ;recebido, retorna uma lista vazia
        ((integer? (/ n i)) ;caso n/i seja inteiro,
         (cons i (f (+ i 1))) ;cria um par contendo i e o resultado de uma função f
        ) ;que envia como parâmetro o valor i+1
        (else (f (+ i 1))) ;caso não seja nenhuma das situações acima, chama a função f
        ) ;que envia como parâmetro o valor i+1
      )
    )
  )
)
```

28. Executar o programa e indicar o seu conteúdo

21h00 – 21h18

Print

```

14-operad-logicos.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias    <===== seu nome aqui e abaixo
;;

#lang racket      ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021")
(newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)")
(newline)
(display " Aluno: João Vitor Fernandes Dias ")
(newline)
;;
;;

(display "Operadores lógicos... ")
(newline)

(let ((x 3))
  (and (> x 2) (< x 4)))

(let ((y 5))
  (and (> y 2) (< y 4)))

(newline)
(display "Tabela de valores AND")
(newline)
(display " T and T = ") (and #t #t)
(newline)
(display " T and F = ") (and #t #f)
(newline)
(display " F and F = ") (and #f #f)
;-----
(newline)
(display "Tabela de valores OR")
(newline)
(display " T or T = ") (or #t #t)

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias
Operadores lógicos...

#t
#f

Tabela de valores AND
T and T = #t
T and F = #f
F and F = #f

Tabela de valores OR
T or T = #t
T or F = #t
F or F = #f

#t
#f

not(5 > 3) = #f
not(10 > 25) = #t
>

```

Explicação

Nesse código é demonstrado o uso dos operadores lógicos AND, OR e NOT em Racket. Começa fazendo alguns testes com a expressão `let` para que, usando o operador AND sejam retornados os valores `#t` e `#f` (`#t` equivale ao valor `true` (verdadeiro) e `#f` equivale ao valor `false` (falso)), posteriormente é mostrada a tabela referente aos valores da tabela-verdade do operador AND, o mesmo se repete para o operador OR (porém, mostrando primeiro a tabela e só depois fazendo os testes com a expressão `let`). Por fim, são testadas duas expressões lógicas utilizando o operador OR, a primeira retornando `#f` e a segunda `#t`.

29. Escreva e teste pelo menos cinco operações lógicas

21h18 – 21h28

Print

```

14-operad-logicos.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

(display "Tabela de valores AND") (newline)
(display " T and T = ") (and #t #t)
(display " T and F = ") (and #t #f)
(display " F and F = ") (and #f #f) (newline)
;-----
(display "Testes com o operador lógico OR") (newline)
;(let ((x 3)) (or (> x 2) (< x 8)))
;(let ((y 10)) (or (> y 20) (< y 7)))

(display "Tabela de valores OR") (newline)
(display " T or T = ") (or #t #t)
(display " T or F = ") (or #t #f)
(display " F or F = ") (or #f #f) (newline)
;-----
(display "Testes com o operador lógico NOT") (newline)
;(display " not( 5 > 3) = " (not (> 5 3)))
;(display " not(10 > 25) = " (not (> 10 25)))

(display "Tabela de valores NOT") (newline)
(display " not T = ") (not #t)
(display " not F = ") (not #f) (newline)
;-----
(display "Tabela de valores ->") (newline)
(define (-> p q) (or (not p) q))
(display " T -> T = ") (-> #t #t)
(display " T -> F = ") (-> #t #f)
(display " F -> T = ") (-> #f #t)
(display " F -> F = ") (-> #f #f) (newline)
;-----
(display "Tabela de valores <->") (newline)
(define (<-> p q) (and (-> p q) (-> q p)))
(display " T <-> T = ") (<-> #t #t)
(display " T <-> F = ") (<-> #t #f)
(display " F <-> T = ") (<-> #f #t)
(display " F <-> F = ") (<-> #f #f)

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

OPERADORES LÓGICOS

Tabela de valores AND
T and T = #t
T and F = #f
F and F = #f

Tabela de valores OR
T or T = #t
T or F = #t
F or F = #f

Tabela de valores NOT
not T = #f
not F = #t

Tabela de valores ->
T -> T = #t
T -> F = #f
F -> T = #t
F -> F = #t

Tabela de valores <->
T <-> T = #t
T <-> F = #f
F <-> T = #f
F <-> F = #t
>|

```

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
; -----
(display "OPERADORES LÓGICOS") (newline) (newline)

;(display "Testes com o operador lógico AND") (newline)
;(let ((x 3)) (and (> x 2) (< x 4)))
;(let ((y 5)) (and (> y 2) (< y 4)))

(display "Tabela de valores AND") (newline)
(display " T and T = ") (and #t #t)
(display " T and F = ") (and #t #f)
(display " F and F = ") (and #f #f) (newline)
;;-----
;(display "Testes com o operador lógico OR") (newline)
;(let ((x 3)) (or (> x 2) (< x 8)))
;(let ((y 10)) (or (> y 20) (< y 7)))

(display "Tabela de valores OR") (newline)
(display " T or T = ") (or #t #t)
(display " T or F = ") (or #t #f)
(display " F or F = ") (or #f #f) (newline)
;;-----
;(display "Testes com o operador lógico NOT") (newline)
;(display " not( 5 > 3) = ") (not (> 5 3))
;(display " not(10 > 25) = ") (not (> 10 25))

(display "Tabela de valores NOT") (newline)
(display " not T = ") (not #t)
(display " not F = ") (not #f) (newline)
;;-----
(display "Tabela de valores ->") (newline)

(define (-> p q) (or (not p) q))
(display " T -> T = ") (-> #t #t)
(display " T -> F = ") (-> #t #f)
(display " F -> T = ") (-> #f #t)
(display " F -> F = ") (-> #f #f) (newline)
;;-----
(display "Tabela de valores <->") (newline)

(define (<-> p q) (and (-> p q) (-> q p)))
(display " T <-> T = ") (<-> #t #t)
(display " T <-> F = ") (<-> #t #f)
(display " F <-> T = ") (<-> #f #t)
(display " F <-> F = ") (<-> #f #f)
```

Print

```

15-predicados.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

(display "4 = 4? ") (eq? 4 4)
(display "car = car? ") (eq? car car)
(display "car = cdr? ") (eq? car cdr)
(display "Falso = Falso? ") (eq? #f #f) (newline)
;; -----
(display "Operador EQUIVALENTE ") (newline)
(display "9/2 é EQUIVALENTE a 7/2? ") (eqv? 9/2 7/2)
(display "12/5 é EQUIVALENTE a 24/10? ") (eqv? 12/5 24/10)
;; -----
(let ((x "Tudo bem?")) (eqv? x x))
;; -----
(boolean? #t)
(boolean? #f)
(display "7 é BOOLEANO? ") (boolean? 7)
;; -----
;; = lista vazia =
(display "() é uma lista NULA? ") (null? '())
(display "(a) é uma lista NULA? ") (null? '(a))
(display "(list 1 3 5 7) é uma lista NULA? ") (null? (list 1 3 5 7))
(display "6 é NULO? ") (null? 6) (newline)
;; -----
(display "Pares? ") (newline)
(display "Par (3,4)? ") (pair? '(3 . 4))
(display "Par 5? ") (pair? 5)
(display "Pares (cons 6 9) ? ") (pair? (cons 6 9))
;; -----
(display "Numero 6.78 ? ") (number? 6.78)
(display "Numero 2/7 ? ") (number? 2) ;(newline)
;; -----
(display "E um número INTEIRO 3 ? ") (integer? 3)
(display "E um número INTEIRO 3.0 ? ") (integer? 3.0)
(display "E um número INTEIRO 3/7 ? ") (integer? 3/7) ;(newline)
;; -----
(display "E um número REAL 2 ? ") (real? 2)
(display "E um número REAL 3/5 ? ") (real? 3/5)
(display "E um número REAL 4+5i ? ") (real? 4+5i) ;(newline)
;; -----
(display "E um número COMPLEXO 5+3i ? ") (complex? 5+3i) ;(newline)
;; -----
(display "E um STRING 'Oi...' ? ") (string? "Oi...")
(display "E um STRING 24 ? ") (string? 24)

```

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UNF-CCT-ICMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

```

Operador IGUAL
4 = 4? #t
car = car? #t
car = cdr? #f
Falso = Falso? #t

Operador EQUIVALENTE
9/2 é EQUIVALENTE a 7/2? #f
12/5 é EQUIVALENTE a 24/10? #t
#t
#t
7 é BOOLEANO? #f
() é uma lista NULA? #t
(a) é uma lista NULA? #f
(list 1 3 5 7) é uma lista NULA? #f
6 é NULO? #f

Pares?
Par (3,4)? #t
Par 5? #f
Pares (cons 6 9) ? #t
Numero 6.78 ? #t
Numero 2/7 ? #t
E um número INTEIRO 3 ? #t
E um número INTEIRO 3.0 ? #f
E um número INTEIRO 3/7 ? #f
E um número REAL 2 ? #t
E um número REAL 3/5 ? #t
E um número REAL 4+5i ? #f
E um número COMPLEXO 5+3i ? #t
E um STRING 'Oi...' ? #t
E um STRING 24 ? #f
>

```

Determine language from source ▼ CRLF 32:56 P 454.27 MB 0 3

Explicação

Este código exemplifica o uso de predicados (?) (não sei se é assim que se deve chamar).

DÚVIDA: O que são de fato os predicados? Expressões com checagem?

Os predicados são expressões de checagem (são mesmo?) que averiguam certa propriedade de um determinado valor. Neste código são mostrados alguns, com alguns exemplos de cada um. Os predicados mostrados foram:

| | |
|----------|--|
| eq? | Confere se os dois valores comparados são iguais |
| eqv? | Confere se os dois valores comparados são equivalentes |
| boolean? | Confere se o valor comparado é booleano |
| null? | Confere se o valor/lista comparado(a) é nulo(a) |
| pair? | Confere se o valor/lista comparado(a) é um par |
| number? | Confere se o valor comparado é um número |
| integer? | Confere se o valor comparado é um número inteiro |
| real? | Confere se o valor comparado é um número real |
| complex? | Confere se o valor comparado é um número complexo |
| string? | Confere se o valor comparado é uma string |

Predicados

(char? 'm)

(char? 14)

(char? #b)

(char? #m)

```

15-predicados-brkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; ----- Lista vazia -----
(display "7 é BOOLEANO? ") (boolean? 7)
;; -----
(display "() é uma lista NULA? ") (null? '())
(display "(a) é uma lista NULA? ") (null? '(a))
(display "(list 1 3 5 7) é uma lista NULA? ") (null? (list 1 3 5 7))
(display "6 é NULO? ") (null? 6) (newline)
;; -----
(display "Pares? ") (newline)
(display "Par (3.4)? ") (pair? '(3 . 4))
(display "Par 5? ") (pair? 5)
(display "Pares (cons 6 9)? ") (pair? (cons 6 9))
;; -----
(display "Número 6.78? ") (number? 6.78)
(display "Número 2/7? ") (number? 2) (newline)
;; -----
(display "É um número INTEIRO 3? ") (integer? 3)
(display "É um número INTEIRO 3.0? ") (integer? 3.0)
(display "É um número INTEIRO 3/7? ") (integer? 3/7) (newline)
;; -----
(display "É um número REAL 2? ") (real? 2)
(display "É um número REAL 3/5? ") (real? 3/5)
(display "É um número REAL 4+5i? ") (real? 4+5i) (newline)
;; -----
(display "É um número COMPLEXO 5+3i? ") (complex? 5+3i) (newline)
;; -----
(display "É um STRING 'Oi...'? ") (string? "Oi...")
(display "É um STRING 24? ") (string? 24) (newline) (newline)
;; ----- Testar os predicados -----
(display "É char? ") (newline)
(display "'m' ") (char? 'm)
(display "14 ") (char? 14)
(display "#\\b ") (char? #\\b)
(display "#\\m ") (char? #\\m)

;; -----
;; Predicados
;; -----
(display "Operador IGUAL ") (newline)
(display "a = 3? ") (eq? 'a 3)
(display "4 = 4? ") (eq? 4 4)
(display "car = car? ") (eq? car car)
(display "car = cdr? ") (eq? car cdr)
(display "Falso = Falso? ") (eq? #f #f) (newline)
;; -----
(display "Operador EQUIVALENTE ") (newline)
(display "9/2 é EQUIVALENTE a 7/2? ") (eqv? 9/2 7/2)
(display "12/5 é EQUIVALENTE a 24/10? ") (eqv? 12/5 24/10)
;; -----
(let ((x "Tudo bem?")) (eqv? x x))
;; -----

```

Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UNF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vítor Fernandes Dias
É char?
'm #f
14 #f
#\\b #t
#\\m #t
>

Código

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)" (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline)
#|
;; -----
;; Predicados
;; -----
(display "Operador IGUAL ") (newline)
(display "a = 3? ") (eq? 'a 3)
(display "4 = 4? ") (eq? 4 4)
(display "car = car? ") (eq? car car)
(display "car = cdr? ") (eq? car cdr)
(display "Falso = Falso? ") (eq? #f #f) (newline)
;; -----
(display "Operador EQUIVALENTE ") (newline)
(display "9/2 é EQUIVALENTE a 7/2? ") (eqv? 9/2 7/2)
(display "12/5 é EQUIVALENTE a 24/10? ") (eqv? 12/5 24/10)
;; -----
(let ((x "Tudo bem?")) (eqv? x x))
;; -----

```



```

(boolean? #t)
(boolean? #f)
(display "7 é BOOLEANO? ") (boolean? 7)
;; ----- Lista vazia -----
(display "() é uma lista NULA? ") (null? '())
(display "(a) é uma lista NULA? ") (null? '(a))
(display "(list 1 3 5 7) é uma lista NULA? ") (null? (list 1 3 5 7))
(display "6 é NULO? ") (null? 6) (newline)
;; -----
(display "Pares? ") (newline)
(display "Par (3.4)? ") (pair? '(3 . 4))
(display "Par 5? ") (pair? 5)
(display "Pares (cons 6 9) ? ") (pair? (cons 6 9))
;; -----
(display "Numero 6.78 ? ") (number? 6.78)
(display "Numero 2/7 ? ") (number? 2) ;(newline)
;; -----
(display "É um número INTEIRO 3 ? ") (integer? 3)
(display "É um número INTEIRO 3.0 ? ") (integer? 3.0)
(display "É um número INTEIRO 3/7 ? ") (integer? 3/7) ;(newline)
;; -----
(display "É um número REAL 2 ? ") (real? 2)
(display "É um número REAL 3/5 ? ") (real? 3/5)
(display "É um número REAL 4+5i ? ") (real? 4+5i) ;(newline)
;; -----
(display "É um número COMPLEXO 5+3i ? ") (complex? 5+3i) ;(newline)
;; -----
(display "É um STRING 'Oi...' ? ") (string? "Oi...")
(display "É um STRING 24 ? ") (string? 24)(newline)(newline)
|#
;; ----- Testar os predicados: -----
(display "É char? ")(newline)
(display "'m ") (char? 'm)
(display " 14 ") (char? 14)
(display "#\b ") (char? #\b)
(display "#\m ") (char? #\m)

```

32. Executar o programa e indicar o que ele faz

Print

The screenshot shows the DrRacket IDE with a Racket program on the left and its output on the right. The program defines several functions and lists, and then uses the `map` function to apply these functions to the lists.

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-OCT-LCMAT - Curso de Ciência da Computação
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
#lang racket ;; define a linguagem default

; -----
(display " UENF-OCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline)

; -----Mapeamentos-----
(define dobro (lambda (x) (* 2 x) ))
(define proximo (lambda (x) (+ 1 x) ))
(define quadrado (lambda (x) (* x x) ))
(define Lista (list 1 4 9 16 25))
(define dados '(2 5 6 23))

;;
(display "Lista: ") Lista
(display "Raiz Lista: ") (map sqrt Lista) (newline)
;;
(display "Dados: ") dados (newline)
;;mapeamentos
(display "Mapeamentos: Dados-Dobro ") (newline) (map dobro dados )
(display "Mapeamentos: Dados-Proximo ") (newline) (map proximo dados )
(display "Mapeamentos: Dados-Quadrado ") (newline) (map quadrado dados )

```

The output on the right shows the results of the program execution:

```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-OCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Lista: '(1 4 9 16 25)
Raiz Lista: '(1 2 3 4 5)
Dados: '(2 5 6 23)

Mapeamentos: Dados-Dobro
'(4 10 12 48)
Mapeamentos: Dados-Proximo
'(3 6 7 24)
Mapeamentos: Dados-Quadrado
'(4 25 36 529)
>

```

Explicação

Este código ilustra o uso da função (?) `map`.

DÚVIDA: “map” é uma função?

A função (?) `map` permite que um procedimento seja efetuado por sobre toda uma lista de dados. Para isso, nesse código foram definidas 3 funções: `dobro`, `proximo` e `quadrado`.

`Dobro` recebe um número e retorna o seu valor multiplicado por dois

`Proximo` recebe um número e retorna o seu valor somado a um

`Quadrado` recebe um número e retorna o seu valor multiplicado a ele mesmo

Não são definidas duas listas sobre as quais será utilizado as funções criadas em conjunto com a função `map`.

33. Construir um NOVO programa que faça o seguinte mapeamento 09h29 – 09h35

Mapeamento

$x \longrightarrow x^2 + 3x - 9$

Print

The screenshot shows the DrRacket IDE interface. The left pane contains a Scheme script with the following content:

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)" (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
; -----Mapeamentos-----
(define aplicarQuadratica (lambda (x) (+ (* x x) (* 3 x) -9) ))
(define lista (list -6 -5 -4 -3 -2 -1.5 -1 0 1 2 3))
;;mapeamentos
(display "Mapeamentos: TesteQuadratica: ") (newline) (map aplicarQuadratica lista )
```

The right pane shows the output of the script:

```
Welcome to DrRacket, version 8.2 [3m]
Language: racket, with debugging; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Mapeamentos: TesteQuadratica:
'(0 1 -5 -9 -11 -11.25 -11 -9 -5 1 9)
>
```

The status bar at the bottom indicates the language is determined from the source, and the memory usage is 483.06 MB.

Código

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)" (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
; -----Mapeamentos-----
(define aplicarQuadratica (lambda (x) (+ (* x x) (* 3 x) -9) ))
(define lista (list -6 -5 -4 -3 -2 -1.5 -1 0 1 2 3))
;;mapeamentos
(display "Mapeamentos: TesteQuadratica: ") (newline) (map aplicarQuadratica lista )
```

Print

```

17-raizes-poly.rkt - DrRacket
File Edit View Language Racket Insert Scripts Tabs Help

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-OCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student

(define (poly2grau a b c)
  (cond
    [(= a 0) 'degenerada]
    [(< (* b b) (* 4 a c)) 'NenhumaOuComplexa]
    [(= (* b b) (* 4 a c)) (/ (- b) (* 2 a))]
    [(> (* b b) (* 4 a c)) (list
      (/ (+ (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
      (/ (- (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
    )]
  )
)

;;----- PrintTexto -----
(define (printText a b c) (begin
  (newline) (newline)
  (display a) (display "X^2 + ")
  (display b) (display "X + ")
  (display c) (display " = 0, Raizes = ")
  (poly2grau a b c)
))

;;-----EXEMPLOS-----
(printText 1 2 1) (display "deveria ser -1")
(printText 3 4 1) (display "deveria ser -1/3 -1")
(printText 2 4 3) (display "deveria ser Nenhuma")
(printText 1 0 -1) (display "deveria ser 1 e -1")
(printText 2 4 2) (display "deveria ser -1")
(printText 0 1 1) (display "deveria ser: Degenerada")

Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student; memory limit: 128 MB.
UENF-OCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

1*X^2 + 2*X + 1 = 0, Raizes = -1
deveria ser -1

3*X^2 + 4*X + 1 = 0, Raizes = (list -0.3 -1)
deveria ser -1/3 -1

2*X^2 + 4*X + 3 = 0, Raizes = 'NenhumaOuComplexa
deveria ser Nenhuma

1*X^2 + 0*X + -1 = 0, Raizes = (list 1 -1)
deveria ser 1 e -1

2*X^2 + 4*X + 2 = 0, Raizes = -1
deveria ser -1

0*X^2 + 1*X + 1 = 0, Raizes = 'degenerada
deveria ser: Degenerada
> |

```

Explicação

Este código ilustra o uso de polinômios. Com a função “polygrau2” que recebe como parâmetro os coeficientes A, B e C de uma equação de segundo grau, ele calcula que tipo de polinômio ele é:

Caso o seu coeficiente A seja igual a zero, ela é degenerada, pois não é uma função de segundo grau.

Caso B^2 seja menor que $4*A*C$, não haverá raiz real.

Caso B^2 seja maior que $4*A*C$, cria uma lista composta pela solução de X1 e X2 utilizando a fórmula de Bhaskara.

(Abaixo eu fiz um procedimento que imprime de forma mais compacta o polinômio com seus coeficientes e também chama a função poly2grau.)

Print

The screenshot shows the DrRacket IDE with two panes. The left pane contains Scheme code for a program that calculates the roots of a quadratic equation $Ax^2 + Bx + C = 0$. The code includes comments in Portuguese and defines a function `poly2grau` that handles different cases based on the discriminant. The right pane shows the output of the program, displaying the equation and its roots for five different sets of coefficients (a, b, c).

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student
;-----
(display " UENF-CCCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; ----Aplicações: raizes do polinomio Ax^2 + Bx + C = 0-----
(define (poly2grau a b c)
  (cond
    [(= a 0) 'degenerada]
    [(< (* b b) (* 4 a c)) 'NenhumaOuComplexa]
    [(= (* b b) (* 4 a c)) (/ (- b) (* 2 a))]
    [(> (* b b) (* 4 a c)) (list
                            (/ (+ (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
                            (/ (- (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
                            )])
  )
)
;;----- PrintTexto -----
(define (printText a b c) (begin
  (newline)(newline)
  (display a) (display "X^2 + ")
  (display b) (display "X + ")
  (display c) (display " = 0, Raizes = ")
  (poly2grau a b c)
))
;;-----EXEMPLOS -----
printText 0 9 9
printText 1 2 3
printText 2 4 2
printText 3 9 3
printText 4 16 4

```

Output:

```

Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student, memory limit: 128 MB.
UENF-CCCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

0*X^2 + 9*X + 9 = 0, Raizes = 'degenerada

1*X^2 + 2*X + 3 = 0, Raizes = 'NenhumaOuComplexa

2*X^2 + 4*X + 2 = 0, Raizes = -1

3*X^2 + 9*X + 3 = 0, Raizes = (list #1-0.3919660112501051
#1-2.618033988749895)

4*X^2 + 16*X + 4 = 0, Raizes = (list #1-0.2679491924311228
#1-3.732050807568877)
>

```

Código

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student
;-----
(display " UENF-CCCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
;; ----Aplicações: raizes do polinomio Ax^2 + Bx + C = 0-----
(define (poly2grau a b c)
  (cond
    [(= a 0) 'degenerada]
    [(< (* b b) (* 4 a c)) 'NenhumaOuComplexa]
    [(= (* b b) (* 4 a c)) (/ (- b) (* 2 a))]
    [(> (* b b) (* 4 a c)) (list
                            (/ (+ (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
                            (/ (- (- b) (sqrt (- (* b b) (* 4 a c)))) (* 2 a))
                            )])
  )
)
;;----- PrintTexto -----
(define (printText a b c) (begin
  (newline)(newline)
  (display a) (display "X^2 + ")
  (display b) (display "X + ")
  (display c) (display " = 0, Raizes = ")
  (poly2grau a b c)
))
;;-----EXEMPLOS -----

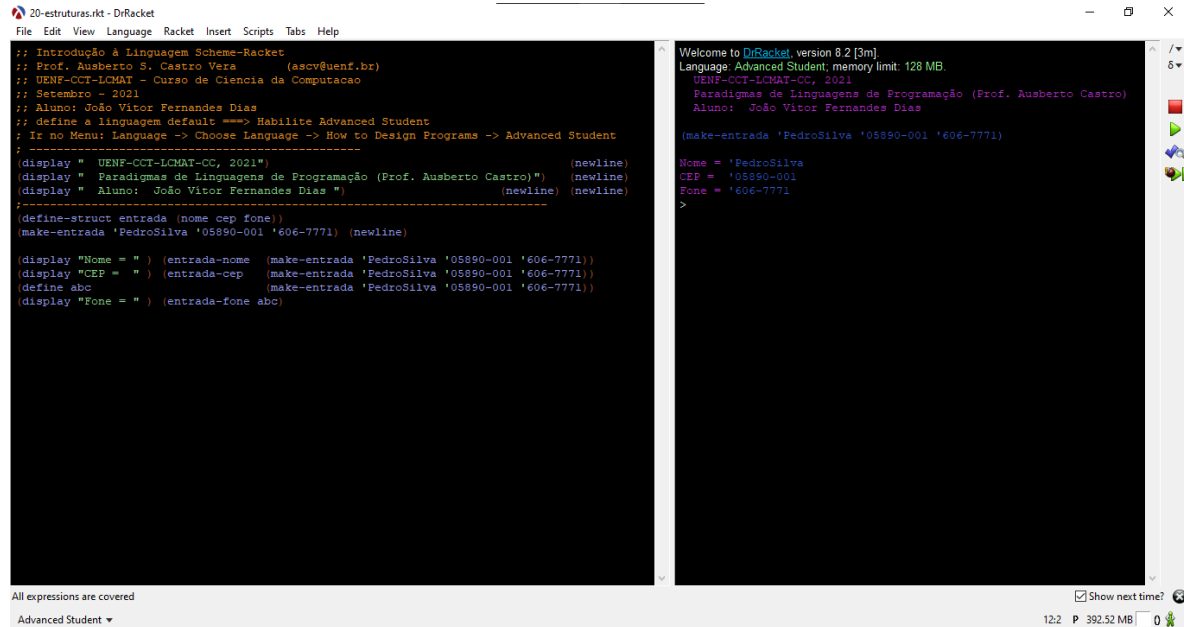
```

```
(printText 0 9 9)
(printText 1 2 3)
(printText 2 4 2)
(printText 3 9 3)
(printText 4 16 4)
```

Arquivo 20-estruturas.rkt Aplicações: Estruturas de dados 10h07 – 10h59

36. Executar o programa e explicar o conteúdo e os resultados 10h07 – 10h32

Print



The screenshot shows the DrRacket IDE with two panes. The left pane contains a Scheme script for a menu-driven program. The right pane shows the output of the program, including a welcome message, menu options, and the results of user input.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student
;; Ir no Menu: Language -> Choose Language -> How to Design Programs -> Advanced Student
;
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias " ) (newline) (newline)
;-----
(define-struct entrada (nome cep fone))
(make-entrada 'PedroSilva '05890-001 '606-7771) (newline)

(display "Nome = " ) (entrada-nome (make-entrada 'PedroSilva '05890-001 '606-7771))
(display "CEP = " ) (entrada-cep (make-entrada 'PedroSilva '05890-001 '606-7771))
(define abc (make-entrada 'PedroSilva '05890-001 '606-7771))
(display "Fone = " ) (entrada-fone abc)
```

Output:

```
Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student; memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

(make-entrada 'PedroSilva '05890-001 '606-7771)

Nome = 'PedroSilva
CEP = '05890-001
Fone = '606-7771
>
```

Explicação

Esse código ilustra o uso de estruturas
A estrutura “entrada” é definida com a função “define-struct” e possuirá 3 campos: nome, cep e fone.
Com a função “make-NomeDaEstrutura” é possível criar uma estrutura
Para exibir apenas um dos campos da estrutura, é necessário utilizar da seguinte forma: “NomeDaEstrutura-NomeDoCampo” seguido da estrutura criada ou de uma função que retorne uma estrutura criada.
E assim é feito utilizando os termos “entrada-nome”, “entrada-cep”, “entrada-fone”.

Print

The screenshot shows the DrRacket IDE with a Scheme program on the left and its output on the right. The program defines a language, sets a default, and creates data structures for an apocalypse simulation. The output shows the results of these definitions and calculations.

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student
;; Ir no Menu: Language -> Choose Language -> How to Design Programs -> Advanced Student
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)
; -----
(define-struct data (dia mes ano))
(define-struct endereco (bairro rua numero))

(define-struct cedulas (duzentos cem cinquenta vinte dez cinco dois um))
(define-struct moedas (cem cinquenta vintecinco dez cinco um))
(define-struct caixa (cedulas moedas))

(define apocalypse (make-data 12 12 2012))
(define casa (make-endereco 'bairro1 'rua1 'numero1))
(define trocador (make-caixa
  (make-cedulas 200 100 50 20 10 5 2 1)
  (make-moedas 1 5 10 25 50 100)
))

(define CedulasDoTrocador (make-cedulas 200 100 50 20 10 5 2 1))
(define MoedasDoTrocador (make-moedas 1 5 10 25 50 100))
(define trocador (make-caixa CedulasDoTrocador MoedasDoTrocador))

(display "Dia do apocalypse = ") (data-dia apocalypse)
(display "Mes do apocalypse = ") (data-mes apocalypse)
(display "Ano do apocalypse = ") (data-ano apocalypse)

(display "Bairro do endereco = ") (endereco-bairro casa)
(display "Rua do endereco = ") (endereco-rua casa)
(display "Numero do endereco = ") (endereco-numero casa)

(display "Cedulas do trocador = ") (caixa-cedulas trocador)
(display "Moedas do trocador = ") (caixa-moedas trocador)

```

Output:

```

Welcome to DrRacket, version 8.2 [3m].
Language: Advanced Student, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Dia do apocalypse = 12
Mes do apocalypse = 12
Ano do apocalypse = 2012
Bairro do endereco = 'bairro1
Rua do endereco = 'rua1
Numero do endereco = 'numero1
Cedulas do trocador = (make-cedulas 200 100 50 20 10 5 2 1)
Moedas do trocador = (make-moedas 1 5 10 25 50 100)
> |

```

Código

```

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera      (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
;; define a linguagem default ==> Habilite Advanced Student
;; Ir no Menu: Language -> Choose Language -> How to Design Programs -> Advanced Student
; -----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline)
; -----
(define-struct data (dia mes ano))
(define-struct endereco (bairro rua numero))

(define-struct cedulas (duzentos cem cinquenta vinte dez cinco dois um))
(define-struct moedas (cem cinquenta vintecinco dez cinco um))
(define-struct caixa (cedulas moedas))

(define apocalypse (make-data 12 12 2012))
(define casa (make-endereco 'bairro1 'rua1 'numero1))
(define trocador (make-caixa
  (make-cedulas 200 100 50 20 10 5 2 1)
  (make-moedas 1 5 10 25 50 100)
))

(define CedulasDoTrocador (make-cedulas 200 100 50 20 10 5 2 1))
(define MoedasDoTrocador (make-moedas 1 5 10 25 50 100))
(define trocador (make-caixa CedulasDoTrocador MoedasDoTrocador))

(display "Dia do apocalypse = ") (data-dia apocalypse)
(display "Mes do apocalypse = ") (data-mes apocalypse)
(display "Ano do apocalypse = ") (data-ano apocalypse)

(display "Bairro do endereco = ") (endereco-bairro casa)
(display "Rua do endereco = ") (endereco-rua casa)
(display "Numero do endereco = ") (endereco-numero casa)

(display "Cedulas do trocador = ") (caixa-cedulas trocador)
(display "Moedas do trocador = ") (caixa-moedas trocador)

```

```
(display "Bairro do endereço = ") (endereco-bairro casa)
(display "Rua do endereço = ") (endereco-rua casa)
(display "Numero do endereço = ") (endereco-numero casa)
```

```
(display "Cedulas do trocador = ") (caixa-cedulas trocador)
(display "Moedas do trocador = ") (caixa-moedas trocador)
```

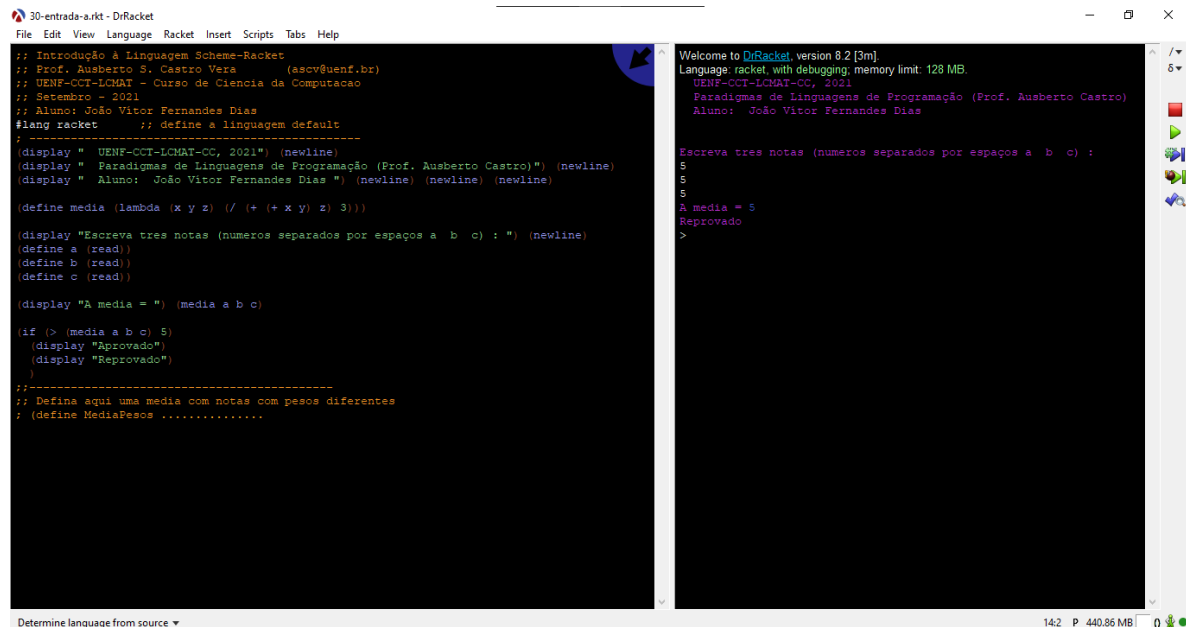
Arquivo 30-entrada.rkt

11h09 – 12H41 DÚVIDA

38. Executar o programa e explicar o conteúdo e os resultados

11h09 – 11h35

Print



The screenshot shows the DrRacket IDE with a file named '30-entrada.rkt'. The editor on the left contains a Racket script that defines a function to calculate the average of three numbers and checks if the average is greater than 5. The output window on the right shows the execution results, including the prompt 'Escreva tres notas' and the calculated average 'A media = 5', followed by the status 'Reprovado'.

```
;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera       (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao
;; Setembro - 2021
;; Aluno: João Vitor Fernandes Dias
#lang racket      ;; define a linguagem default
;-----
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline) (newline)

(define media (lambda (x y z) (/ (+ x y z) 3)))

(display "Escreva tres notas (numeros separados por espacos a b c) : ") (newline)
(define a (read))
(define b (read))
(define c (read))

(display "A media = ") (media a b c)

(if (> (media a b c) 5)
    (display "Aprovado")
    (display "Reprovado"))

;-----
;; Defina aqui uma media com notas com pesos diferentes
; (define MediaPesos .....
```

```
Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Escreva tres notas (numeros separados por espacos a b c) :
5
5
5
A media = 5
Reprovado
>
```

Explicação

Esse código exemplifica o uso da entrada de dados pelo usuário. Utiliza a função `read` para permitir tal feito.

Esse código define a função de média, em seguida define os valores das notas A, B e C de acordo com o input do usuário, calcula a média com a função previamente definida, mostra seu valor e então confere se essa média é maior do que 5, se for, “Aprovado” será mostrado na tela, senão, “Reprovado” será mostrado.

39. Criar um programa NOVO que faça a leitura de dados pessoais de duas pessoas (utilize entrada de dados)

DÚVIDA

11h35 – 12h41

Print

The screenshot shows the DrRacket IDE with a Racket program on the left and an error message on the right. The program defines a structure for personal data and a procedure to read it. The error message states: "define: not allowed in an expression context in: (define d (read))".

```
(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vitor Fernandes Dias ") (newline) (newline)

(define-struct data (dia mes ano))
(define-struct endereco (bairro rua numero))
(define-struct dadoPessoal (data endereco))

(define (mostrarDadosPessoais pessoa) (begin (
;geral
(display "Data de nascimento: ") (display dadoPessoal-data)
(display "Endereço: ") (display dadoPessoal-endereco)
(newline) (newline)
;Mais especifico
;(display "Data de nascimento: ") (display dadoPessoal-data-dia)
;(display "/" ) (display dadoPessoal-data-mes)
;(display "/" ) (display dadoPessoal-data-ano)
;(display " ") (newline)

;(display "Endereço: (Bairro: ") (display dadoPessoal-endereco-bairro)
;(display "; Rua: ") (display dadoPessoal-endereco-rua)
;(display "; Número: ") (display dadoPessoal-endereco-numero)
;(display " ") (newline)
))

(define lerDadosPessoais (begin (
(display "Digite sua data de nascimento: ") (newline)

(display "Dia: ") (define d (read))
(display "Mês: ") (define m (read))
(display "Ano: ") (define a (read))
(define nascimento (make-data d m a))

(display "Digite seu endereço: ") (newline)
(display "Bairro: ") (define b (read))
(display "Rua: ") (define r (read))
(display "Número: ") (define n (read))

(define casa (make-endereco b r n))

(define pessoa (make-dadoPessoal casa nascimento))
(mostrarDadosPessoais pessoa)
))

(lerDadosPessoais)
(lerDadosPessoais)
```

30-entrada-c.rkt:36:22: define: not allowed in an expression context in: (define d (read))

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Define: not allowed in an expression context in: (define d (read))

DÚVIDA: Por que o define não está funcionando?

Utilizei a biblioteca Block e removi uns parênteses

Mas...

The screenshot shows the DrRacket IDE with a Racket program on the left and an error message on the right. The program uses the 'block' library to read input. The error message states: "application: not a procedure; expected a procedure that can be applied to arguments".

```
(newline) (newline)
;Mais especifico
;(display "Data de nascimento: ") (display dadoPessoal-data-dia)
;(display "/" ) (display dadoPessoal-data-mes)
;(display "/" ) (display dadoPessoal-data-ano)
;(display " ") (newline)

;(display "Endereço: (Bairro: ") (display dadoPessoal-endereco-bairro)
;(display "; Rua: ") (display dadoPessoal-endereco-rua)
;(display "; Número: ") (display dadoPessoal-endereco-numero)
;(display " ") (newline)
))

(define lerDadosPessoais (block
(display "Digite sua data de nascimento: ") (newline)

(display "Dia: ") (define d (read))
(display "Mês: ") (define m (read))
(display "Ano: ") (define a (read))
(define nascimento (make-data d m a))

(display "Digite seu endereço: ") (newline)
(display "Bairro: ") (define b (read))
(display "Rua: ") (define r (read))
(display "Número: ") (define n (read))
(define casa (make-endereco b r n))

(define pessoa (make-dadoPessoal casa nascimento))
(mostrarDadosPessoais pessoa)
))

(lerDadosPessoais)
(lerDadosPessoais)
```

Welcome to DrRacket, version 8.2 [3m].
Language: racket, with debugging, memory limit: 128 MB.
UENF-CCT-LCMAT-CC, 2021
Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)
Aluno: João Vitor Fernandes Dias

Digite sua data de nascimento:
Dia: 12
Mês: 12
Ano: 2012
Digite seu endereço:
Bairro: bairrinho
Rua: ruinha
Número: 12
Data de nascimento: #<procedure:dadoPessoal-data>Endereço:
#<procedure:dadoPessoal-endereco>

application: not a procedure;
expected a procedure that can be applied to arguments
given: #<void>
arguments...: (none)

DÚVIDA: Qual é o problema agora?

Código

;; Introdução à Linguagem Scheme-Racket
;; Prof. Ausberto S. Castro Vera (ascv@uenf.br)
;; UENF-CCT-LCMAT - Curso de Ciencia da Computacao

```

;; Setembro - 2021
;; Aluno: João Vítor Fernandes Dias
#lang racket ;; define a linguagem default
(require racket/block)
; -----

(display " UENF-CCT-LCMAT-CC, 2021") (newline)
(display " Paradigmas de Linguagens de Programação (Prof. Ausberto Castro)") (newline)
(display " Aluno: João Vítor Fernandes Dias ") (newline) (newline) (newline)

(define-struct data (dia mes ano))
(define-struct endereco (bairro rua numero))
(define-struct dadoPessoal (data endereco))

(define (mostrarDadosPessoais pessoa) (block
;geral
(display "Data de nascimento: ") (display dadoPessoal-data)
(display "Endereço: ") (display dadoPessoal-endereco)
(newline) (newline)
;Mais específico
;(display "Data de nascimento: ") (display dadoPessoal-data-dia)
;(display "/" ) (display dadoPessoal-data-mes)
;(display "/" ) (display dadoPessoal-data-ano)
;(display " ") (newline)

;(display "Endereço: (Bairro: ") (display dadoPessoal-endereco-bairro)
;(display "; Rua: ") (display dadoPessoal-endereco-rua)
;(display "; Número: ") (display dadoPessoal-endereco-numero)
;(display " ") (newline)
))

(define lerDadosPessoais (block
(display "Digite sua data de nascimento: ") (newline)

(display "Dia: ") (define d (read))
(display "Mês: ") (define m (read))
(display "Ano: ") (define a (read))
(define nascimento (make-data d m a))

(display "Digite seu endereço: ") (newline)
(display "Bairro: ") (define b (read))
(display "Rua: ") (define r (read))
(display "Número: ") (define n (read))
(define casa (make-endereco b r n))

(define pessoa (make-dadoPessoal casa nascimento))

(mostrarDadosPessoais pessoa)
))

(lerDadosPessoais)
(lerDadosPessoais)

```