Concurso Engenharia de Computação – Ponto 3
Gabarito

Marcelo Rodrigues de Sousa

15 de março de 2016

Questão 1: Representação de circuitos

```
(defun bits(n b &optional (acc NIL))
  (if (= n 0) acc
      (bits (- n 1) (ash b -1) (cons (oddp b) acc))
  );;end cond
);;end defun

(defun nbits(n b &optional (acc NIL))
  (cond ( (= n 0) acc)
( (oddp b) (nbits (- n 1) (ash b -1) (cons 1 acc)))
( t (nbits (- n 1) (ash b -1) (cons 0 acc)))
  );;end cond
);;end defun

(defun runtable(fn b n m)
  (format t "~{~a ~} ~a~%"
       (nbits b n)
       (if (apply fn (bits b n)) 1 0))
  (when (< (+ n 1) m) (runtable fn b (+ n 1) m) ))

(defmacro table(wire circuit)
  `(runtable ,(eval (list 'lambda wire circuit))
     ,(length wire)
     0
     ,(expt 2 (length wire)) ))

CL-USER(1): (load "ttable")
T
CL-USER(2): (table (x y) (or (and (not x) y) (and x (not y))))
0 0  0
0 1  1
1 0  1
1 1  0
NIL
```


4.

Questão 2: loop

Visitar cada elemento de uma lista

```
CL-USER(2): (loop for i in '(1 2 3) do (prin1 i))
123
```

Visitar cada elemento de um vetor

```
CL-USER(3): (loop for i across #(4 5 6)
                  do (format t "~a " i))
4 5 6
```

Visitar cada chave/valor de uma hashtable

```
CL-USER(5): (defvar h1)
H1

CL-USER(6): (setf h1 (make-hash-table))
#<HASH-TABLE :TEST EQL :COUNT 0 {1004D9ED03}>

CL-USER(7): (setf (gethash 'edu h1) "ed@mort.org")
"ed@mort.org"

CL-USER(8): (setf (gethash 'nero h1) "nerone@roma.gov")
"nerone@roma.gov"
CL-USER(9): (loop for k being the hash-key
                  using (hash-value v) of h1
                  do (format t "~a ~a~%" k v))
EDU ed@mort.org
NERO nerone@roma.gov
```

Executar loop enquanto uma condição persistir

```
CL-USER(10): (loop for i from 1 to 5 while (< i 3) do (print i))
1
2
```

Acumular os elementos de uma lista com collect

```
CL-USER(10): (loop for i from 1 to 5 while (< i 3) collect i)
(1 2)
```

Questão 3: macro de repetição

No programa source abaixo, mostra-se o macro solicitado e um exemplo de execução em comentários.

```
;; File: rec.lisp

(defun getvars(r)
  (if (equal (car r) 'in) NIL
    (cons (list (car r) (cadr r))
  (getvars (cddr r)) )) )

(defun dropvars(r)
  (cond ( (null r)
 (error "(rec g x vx y vy in ...)"))
( (equal (car r) 'in) (cdr r))
(t (dropvars (cddr r)))  ))

(defmacro rec (nm &rest r)
  (let* ( (body (dropvars r))
  (vs (getvars r)))
    `(labels ( (,nm ,(mapcar #'car vs) ,@body))
       (,nm ,@(mapcar #'cadr vs))  )))

(defun fib(n)
  (rec again
       i n
       fi 1
       fi-1 1 in
       (if (< i 2) fi
 (again (- i 1) (+ fi fi-1) fi))  ))

;;Exemplo de uso:
 * (load "rec.lisp")
 T
 * (fat 5)
 120
 * (fat 8)
 40320
```

Questão 4: CLOS

```
(defclass pnt ()
  ( (px :accessor x
:reader getx
:writer setx
:initarg :x)
    (py :accessor y
:reader gety
:writer sety
:initarg :y)))

(defparameter p1 (make-instance 'pnt :x 32 :y 45))
(defparameter p2 (make-instance 'pnt :x 42 :y 55))

(defmethod print-object((p pnt) stream)
  (format stream "{x=~a, y=~a}" (getx p) (gety p)))

(defgeneric add ()
  (:documentation "It adds two objects."))

(defmethod add((p pnt) (q pnt))
  (make-instance 'pnt
 :x (+ (getx p) (getx q))
 :y (+ (gety p) (gety q)) ))

CL-USER(4): (load "clos")
T
CL-USER(5): p1

{x=32, y=45}
CL-USER(6): p2

{x=42, y=55}
CL-USER(7): (add p1 p2)

{x=74, y=100}
```

Questão 5: Usando orgmode

Monte o cabeçalho e aperte TAB na última coluna. Assim:

Nome feet inches meters (aperte TAB aqui)

Os principais comandos para manipular tabelas são:

- TAB leva cursor para a coluna seguinte
- TAB na última coluna: abre uma nova linha
- C-c C-c reformata a tabela
- C-c + soma uma coluna da tabela
- C-y introduz o resultado da soma to texto

Name	Feet	Inches	meters
Pope	4	6	1.3716
Napoleon	5	2	1.5748

Fórmulas

C-c = introduz uma fórmula na coluna do cursor. Se você quiser ângulos em radianos, acrescente um R à expressão. Por exemplo, integ($1,t);R acha a integral da coluna $1 com ângulos em rad. $1 e $2 indicam as colunas 1 e 2 respectivamente.

Aceleração	Velocidade	Posição
a	a t + v	a t^2 / 2 + t v
r*sin(w*t)	-(r cos(t w) / w)	-(r sin(t w) / w^2)

Para separar o cabeçalho dos dados por uma linha horizontal, digite o símbolo |- e aperte tab

Aceleração	Velocidade	Posição

Programas

Um engenheiro civil constrói casas. Um engenheiro de computação constrói programas, robôs e outras criaturinhas adoráveis.

ID	Feet	Inches	meters
Pope	4	6	1.3716
Napoleon	5	2	1.5748
Alexander	5	0	1.524
Schubert	5	1.5	1.5621

Um programa deve residir em um bloco que começa com a palavra chave **BEGIN_SRC** e termina com **END_SRC**. Para executá-lo, faz-se C-c C-c dentro do bloco.

```
                          ; Para executar o programa,
(defun avg(s)             ; aperte C-c C-c dentro do bloco
  (loop for x in s
summing x into sm
counting x into n
    finally (return (/ sm n)) ))

(format nil  "Média  - ~s"  (avg table))
```


4.

Questão 6: Processos

Será chamado um processo em C, que imprime Hello, World. Para tal, inicialmente, far-se-á a tangling do programa em C.

```
#include <stdio.h>
int main() {
   printf("Hello");
   return 1;
}
```

```
(defun chama-media (m p)
  (interactive "r")
  (call-process-region
  m  ;;start of region
  p  ;;end of region
  "~/grt/xhello"  ;;the program
  t ;; when t deletes region before inserting result
  (current-buffer)  ;;destination
  t  ;; redisplay during output
  ;; arguments required by the process
  ""
  "" ))

(global-set-key (kbd "C-c d") 'chama-media)
```

Depois de compilar o programa C com o nome xhello, basta teclar C-c d para obter a resposta no texto.

```
 Como compilar um programa em C
~/grt $ gcc hello.c -o xhello
~/grt $ ./xhello
Hello~/grt $
```

Testando um processo

Depois de compilado o programa em C, necessita-se de carregar o lado do emacs do gerenciador de processos.

M-x load-file

No minibuffer, digite callProcess.el. Para testar, marque uma região e aperte

C-c d

Hello

Questão 7: Internet Scripts

```
!/home/strue028/bin/sbcl.xxx --script

(defun inss(wages)
  (let ((x (if (< wages 5000.0) wages 5000.0)))
     (* 0.1 x)))

(defun getnum(x)
  (let ( (xx (cl-ppcre::all-matches-as-strings
     "-?[0-9]+\\.?[0-9]*" x)))
    (if xx
       (read-from-string (first xx))
100)))

(format t "Content-type: text/html~%~%")

(format t
"<html>
  <head>
    <title>INSS</title>
    <meta http-equiv='Content-Type'
       content='text/html; charset=utf-8'/>
  </head>
  <body>
    <h1>Calculando o INSS</h1>
    <form name='form' method='get'>
       <input type='type='text' name='txt_sal'>
       <input type='submit' name='btn_calcular' value='Send'>
    </form>
    <p>INSS: ~a</p>
  </body>
</html>"

  (inss (getnum (sb-unix::posix-getenv "QUERY_STRING")))
)
```

Questão 8: format

Determinando a diretiva correta para imprimir listas como solicitado.

```
CL-USER(1): (format nil "[~{~a~^,~}]" '(a b c d))

"[A,B,C,D]"
```

Respondendo o resto da questão, cria-se um arquivo "test.dat"contendo uma lista de números.

```
(defun wrt(pth data)
   (with-open-file
       (W pth :direction :output
      :if-exists :supersede)
     (format W "~a ~%" data)  ))

(defun bintable(f-in)
  (with-open-file (in f-in)
   (with-open-file (W #P"out.txt"
  :direction :output
  :if-exists :supersede)
       (dolist (x (read in))
  (format W "~4a => ~8B ~%" x  x)) )))
```

Depois de fazer o tangling com C-c C-v t, testa-se o programa.

```
CL-USER(3): (load "bin.lisp")

T
CL-USER(4): (wrt "test.dat" '(5 7 12 9))

NIL

CL-USER(7): (bintable "test.dat")

NIL
CL-USER(10): (exit)
bash-3.2$ cat out.txt
5    =>      101
7    =>      111
12   =>     1100
9    =>     1001
```

Questão 9: Lendo arquivo

```
(ql:quickload :cl-ppcre)

(defun words(str)
  (cl-ppcre:split "\\s+|,\\s*|\\.\\s*" str))

(defun inputLines(arquivo)
  (with-open-file (stm arquivo :external-format :utf-8)
     (loop for line = (read-line stm nil 42)
 for i = 1 then (+ i 1)
       until (eq line 42)
       collect (list i (words line))  )))
```

Saída

Entrando no eshell, chama-se a sbcl e executa-se a função inputLines no arquivo varela.txt. A interação na REPL da lisp é apresentada abaixo.

```
CL-USER(1): (load "rdlines")
To load "cl-ppcre":
  Load 1 ASDF system:
    cl-ppcre
; Loading "cl-ppcre"
..
T
CL-USER(2): (inputLines "varela.txt")

((1 ("Eras" "na" "vida" "a" "pomba" "predileta"))
 (2 ("que" "sobre" "um" "mar" "de" "ang?stias" "conduzia"))
 (3 ("o" "ramo" "da" "esperan?a" "Eras" "a" "estrela"))
 (4 ("que" "entre" "as" "nevoas" "do" "inverno" "cintilava"))
 (5 ("apontando" "o" "caminho" "ao" "pegureiro")))
```


4.

Questão 10: Concordância

```
(ql:quickload :cl-ppcre)

(defun rdfile(arquivo)
  (with-open-file (stm arquivo :external-format :utf-8)
     (loop for line = (read-line stm nil 42)
 for i = 1 then (+ i 1)
       until (eq line 42)
       append (mapcar (lambda(x) (list x i))
       (cl-ppcre:split
"\\s+|,\\s*|\\.\\s*" line))) ))

(defun srt(arq)
  (sort (rdfile arq) (lambda(x y) (string-lessp (car x) (car y))) ))

(defun words(str)
  (cl-ppcre:split "\\s+|,\\s*|\\.\\s*" str))
```

Resposta da questão 2

Essa questão é semelhante à primeira. A diferença é que, no loop, utiliza-se append em vez de collect.

```
CL-USER(3): (load "concord")
T
CL-USER(4): (srt "varela.txt")

(("a" 1) ("a" 3) ("ang?stias" 2) ("ao" 5) ("apontando" 5) ("as" 4)
 ("caminho" 5) ("cintilava" 4) ("conduzia" 2) ("da" 3) ("de" 2) ("do" 4)
 ("entre" 4) ("Eras" 1) ("Eras" 3) ("esperan?a" 3) ("estrela" 3) ("inverno" 4)
 ("mar" 2) ("na" 1) ("nevoas" 4) ("o" 3) ("o" 5) ("pegureiro" 5) ("pomba" 1)
 ("predileta" 1) ("que" 2) ("que" 4) ("ramo" 3) ("sobre" 2) ("um" 2) ("vida" 1))
```


Marcelo Rodrigues de Sousa
Presidente