Instituto Superior Técnico

Lic. em Matemática Aplicada e Computação
Mestrado Integrado em Eng. Biomédica

Elementos de Programação

19 de Janeiro de 2018 Exame 1 Duração: 2h30

Grupo I (5 valores)

Sabe-se que para todo o número inteiro positivo n existem inteiros não-negativos i, j, k que são *palíndromos* (na habitual notação decimal) tais que $n = i + j + k$. Por exemplo, tem-se $31415926 = 31400413 + 15251 + 262$.

Defina imperativamente em *Python* uma função `threepals` que dado um inteiro positivo `n` devolva um triplo de palíndromos `(i,j,k)` cuja soma seja `n`. Pode usar, sem a definir, uma função auxiliar `palQ` que dado um número natural devolve `True` se se tratar de um palíndromo, e `False` caso contrário.

Resolução:

```python
def threepals(n):
    ok=False
    i=0
    while not(ok):
        if palQ(i):
            j=0
            while not(ok) and j+i<=n:
                if palQ(j) and palQ(n-i-j):
                    ok=True
                else:
                    j=j+1
        if not(ok):
            i=i+1
    return (i,j,n-i-j)
```

Grupo II (4+4 valores)

...				0	0	1	0	1				...

Considere o tipo de dados *tape* que consiste de uma fita de memória infinita dividida em células, onde cada célula pode conter um símbolo (no exemplo temos 0, 1 e células vazias), equipada com uma cabeça de leitura/escrita (cuja posição é indicada na figura pela célula sombreada). Identificaram-se as seguintes operações:

- `new()`: *tape* vazia (todas as células estão vazias);
- `left(t)`: *tape* que resulta de `t` movendo a cabeça de leitura/escrita para a célula imediatamente à esquerda;
- `right(t)`: *tape* que resulta de `t` movendo a cabeça de leitura/escrita para a célula imediatamente à direita;
- `blankQ(t)`: `True` se e só se está vazia a célula de memória onde está posicionada a cabeça de leitura/escrita de `t`;
- `read(t)`: símbolo que está em `t` na célula de memória onde está posicionada a cabeça de leitura/escrita;
- `write(t,s)`: *tape* que resulta de `t` escrevendo o símbolo `s` na célula de memória onde está posicionada a cabeça de leitura/escrita;
- `tbfQ(e)`: `True` se e só se `e` é uma *tape* bem formada.

Por exemplo, sendo `t` a *tape* indicada acima, `left(write(left(t),1))` deverá resultar na *tape*

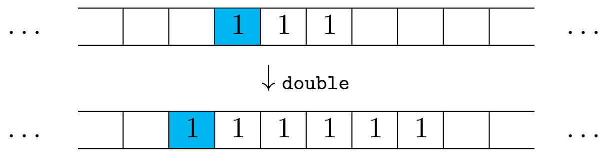
...				1	0	1	0	1				...

a) Desenvolva em *Python* uma implementação eficiente deste tipo de dados, de modo a que cada *tape* seja representada por uma lista da forma $[\mathtt{p}, (\mathtt{p_1}, \mathtt{s_1}), ..., (\mathtt{p_n}, \mathtt{s_n})]$ onde `p` é um inteiro que identifica a posição actual da cabeça de leitura/escrita, cada $\mathtt{p_i}$ é um inteiro que identifica a posição de uma célula não vazia e $\mathtt{s_i}$ é o símbolo nela inscrito. Convenciona-se que as células são numeradas sequencialmente e que `0` corresponde à posição original da cabeça de leitura/escrita no momento de criação da *tape*).

Resolução:

```
def new():
    return [0]

def left(t):
    t[0]=t[0]-1
    return t

def right(t):
    t[0]=t[0]+1
    return t

def blankQ(t):
    return t[0] not in [p[0] for p in t[1:]]

def read(t):
    return [p[1] for p in t[1:] if p[0]==t[0]][0]

def write(t,s):
    return [t[0]]+[p for p in t[1:] if p[0]!=t[0]]+[(t[0],s)]

def tbfQ(e):
    pairsok=all([(type(p) is tuple) and len(p)==2 and (type(p[0]) is int) for p in e[1:]])
    nodups=all([p[0]!=q[0] or p[1]==q[1] for p in e[1:] for q in e[1:]])
    return (type(e) is list) and len(e)>0 and (type(e[0]) is int) and pairsok and nodups
```

b) Desenvolva em *Python*, sobre a camada de abstracção acima desenvolvida e assegurando a independência da implementação, uma função `double` que, recebendo uma *tape* cuja cabeça de leitura/escrita está colocada na célula mais à esquerda de uma sequência finita de 1s (com todas as outras células vazias), devolve a *tape* alterada por forma a que no final a sua cabeça de leitura/escrita esteja colocada na célula mais à esquerda de uma sequência de 1s com o dobro do tamanho.



Resolução:

```python
def double(t):
    n=0
    while not(blankQ(t)):
        n=n+1
        t=right(t)
    while n!=0:
        n=n-1
        t=write(t,1)
        t=right(t)
    left(t)
    while not(blankQ(t)):
        t=left(t)
    t=right(t)
    return t
```

Grupo III (4 valores)

Neste exercício não pode usar recursão, ciclos ou atribuições, nem *strings*. Pode usar, sem necessitar de os definir, os combinadores `map`, `reduce`, `any`, `all`, `filter`, `nest`, `fixedpoint`, bem como definições `lambda` e por compreensão.

Considere o seguinte programa em *Python*.

```
def f(w):
    def g(i):
        if 2*i>len(w) or w[i-1]!=w[len(w)-i]:
            return i
        else:
            return i+1

    return 2*fixedpoint(g,1)>len(w)
```

Implemente funcionalmente em *Python*, tirando partido da função `f` definida acima, a função `palQ` usada no **Grupo I**.

Resolução:

Como `f` verifica se uma dada lista é uma capicua, um número será um palíndromo se a lista dos seus dígitos for aceite por `f`.

```
def diglist(n):
    def trunc(pair):
        if pair[0]==0:
            return pair
        else:
            return (pair[0]//10,[pair[0]%10]+pair[1])
    return fixedpoint(trunc,(n,[]))[1]

def palQ(n):
    return f(diglist(n))
```

Grupo IV (3 valores)

Considere o seguinte programa imperativo PROG.

```
i=len(w)
r=0
while i!=0:
    i=i-1
    r=w[i]+x*r
```

Demonstre que é válida a asserção

$$\{\texttt{True}\}\ \texttt{PROG}\ \{\texttt{r} == \sum_{0 \le \texttt{j} < \texttt{len(w)}} \texttt{w[j]} * \texttt{x}^{\texttt{j}}\}.$$

Resolução: Considera-se a estrutura de ciclo inicializado

$$\texttt{PROG}\left\{\begin{array}{l} \left.\begin{array}{l}\texttt{i} = \texttt{len(w)} \\ \texttt{r} = 0\end{array}\right\}\texttt{INIC} \\ \left.\begin{array}{l}\texttt{while i} != 0: \\ \quad\left.\begin{array}{l}\texttt{i} = \texttt{i} - 1 \\ \texttt{r} = \texttt{w[i]} + \texttt{x} * \texttt{r}\end{array}\right\}\texttt{PASSO}\end{array}\right\}\texttt{CICLO}\end{array}\right.$$

e a condição invariante do ciclo

$$C_{\text{inv}} \equiv (\texttt{r} == \sum_{\texttt{i} \le \texttt{j} < \texttt{len(w)}} \texttt{w[j]} * \texttt{x}^{\texttt{j}-\texttt{i}}).$$

A demonstração da asserção segue abaixo, onde surgem a vermelho justificações para a validade das condições Booleanas.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\color{red}{\mathtt{w}[\mathtt{i}-1] + \mathtt{x} * \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}} == \mathtt{w}[\mathtt{i}-1] * \mathtt{x}^{0} + \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}+1} == \sum_{\mathtt{i}-1\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-(\mathtt{i}-1)}}}
      {(C_{\text{inv}}\ \mathtt{and}\ \mathtt{i} \mathrel{!}= 0) \Rightarrow \mathtt{w}[\mathtt{i}-1] + \mathtt{x} * \mathtt{r} == \sum_{\mathtt{i}-1\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}+1}}\,(\textbf{Rval})
      \quad
      \dfrac{}{\{\mathtt{w}[\mathtt{i}-1] + \mathtt{x} * \mathtt{r} == \sum_{\mathtt{i}-1\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}+1}\}\ \mathtt{i} = \mathtt{i}-1\ \{\mathtt{w}[\mathtt{i}] + \mathtt{x} * \mathtt{r} == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}}\,(\textbf{Ratr})
    }{\{C_{\text{inv}}\ \mathtt{and}\ \mathtt{i} \mathrel{!}= 0\}\ \mathtt{i} = \mathtt{i}-1\ \{\mathtt{w}[\mathtt{i}] + \mathtt{x} * \mathtt{r} == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}}\,(\textbf{Rpre})
    \quad
    \dfrac{}{\{\mathtt{w}[\mathtt{i}] + \mathtt{x} * \mathtt{r} == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}\ \mathtt{r} = \mathtt{w}[\mathtt{i}] + \mathtt{x} * \mathtt{r}\ \{C_{\text{inv}}\}}\,(\textbf{Ratr})
  }{
    \dfrac{\{C_{\text{inv}}\ \mathtt{and}\ \mathtt{i} \mathrel{!}= 0\}\ \color{blue}{\mathtt{PASSO}}\ \{C_{\text{inv}}\}}
    {\{C_{\text{inv}}\}\ \color{blue}{\mathtt{CICLO}}\ \{C_{\text{inv}}\ \mathtt{and\ not}\ (\mathtt{i} \mathrel{!}= 0)\}}\,(\textbf{Riter})
  }\,(\textbf{Rseq})
  \quad
  \dfrac{\color{red}{\text{se } \mathtt{i} == 0 \text{ então } \mathtt{r} == \sum_{0\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-0}}}
  {(C_{\text{inv}}\ \mathtt{and\ not}(\mathtt{i} \mathrel{!}= 0)) \Rightarrow \mathtt{r} == \sum_{0\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}}}\,(\textbf{Rval})
}{\{C_{\text{inv}}\}\ \color{blue}{\mathtt{CICLO}}\ \{\mathtt{r} == \sum_{0\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}}\}}\,(\textbf{Rpos})
$$

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{\color{red}{\sum_{\emptyset} \ldots \text{ é } 0}}
      {\mathtt{True} \Rightarrow 0 == \sum_{\mathtt{len(w)}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{len(w)}}}\,(\textbf{Rval})
      \quad
      \dfrac{}{\{0 == \sum_{\mathtt{len(w)}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{len(w)}}\}\ \mathtt{i} = \mathtt{len(w)}\ \{0 == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}}\,(\textbf{Ratr})
    }{\{\mathtt{True}\}\ \ \mathtt{i} = \mathtt{len(w)}\ \{0 == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}}\,(\textbf{Rpre})
    \quad
    \dfrac{}{\{0 == \sum_{\mathtt{i}\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}-\mathtt{i}}\}\ \mathtt{r} = 0\ \{C_{\text{inv}}\}}\,(\textbf{Ratr})
  }{\{\mathtt{True}\}\ \color{blue}{\mathtt{INIC}}\ \{C_{\text{inv}}\}}\,(\textbf{Rseq})
  \quad
  \dfrac{\vdots}{\{C_{\text{inv}}\}\ \color{blue}{\mathtt{CICLO}}\ \{\mathtt{r} == \sum_{0\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}}\}}
}{\{\mathtt{True}\}\ \color{blue}{\mathtt{PROG}}\ \{\mathtt{r} == \sum_{0\leq \mathtt{j}<\mathtt{len(w)}} \mathtt{w}[\mathtt{j}] * \mathtt{x}^{\mathtt{j}}\}}\,(\textbf{Rseq})
$$