Instituto Superior Técnico

Lic. em Matemática Aplicada e Computação

Mestrado Integrado em Eng. Biomédica

Elementos de Programação

Dezembro de 2016 Exame tipo Duração: 2h30

Grupo I (5 valores)

Neste exercício não pode usar definições por compreensão. As únicas operações sobre listas permitidas são: lista vazia (`[]`), acesso aos elementos da lista por posição (`lis[pos]`), seccionamento da lista (`lis[pos:pos]`), concatenação (`+`) e cálculo do comprimento (`len`). Pode usar `range`.

Defina imperativamente em *Python* uma função `coberturaQ` que dado um número `x` e uma lista de números `w` devolva `True` se existe $I \subseteq \{0, \ldots, \texttt{len(w)} - 1\}$ tal que $\sum_{\texttt{i} \in I} \texttt{w[i]}$ é `x`, e `False` caso contrário.

Nomeadamente, `coberturaQ(11,[4,2,3,10,6])` deverá ser `True` (pois `2+3+6` é `11`), mas `coberturaQ(11,[2,3,10])` deverá ser `False`.

Grupo II (4+4 valores)

Considere filas de espera com tolerâncias, em que cada elemento entra na fila com uma dada tolerância (um número inteiro positivo), que decrementa sempre que sai um elemento da fila, e que faz com que abandone a fila ao chegar a 0, com as seguintes operações:

- `vazia`: fila vazia;
- `entra(f,x,t)`: fila que resulta da entrada na fila `f` do elemento `x` com tolerância `t`;
- `prox(f)`: elemento que está no início da fila `f`;
- `sai(f)`: fila que resulta da saída do próximo elemento de `f`;
- `compr(f)`: número de elementos de `f`;
- `ftbfQ(e)`: `True` se e só se `e` é uma fila com tolerâncias bem formada.

Note, por exemplo, que `sai(entra(entra(entra(vazia(),x,1),y,1),z,2))` deverá ser igual a `entra(vazia(),z,1)`.

a) Desenvolva em *Python* uma implementação eficiente deste tipo de dados, de modo a que cada fila com tolerâncias seja representada por uma lista da forma $[(\texttt{x}_1, \texttt{t}_1), ..., (\texttt{x}_\texttt{n}, \texttt{t}_\texttt{n})]$ onde os elementos surgem por ordem de saída,

isto é, cada x_i é o i-ésimo elemento que sairá da fila (se a sua tolerância t_i o permitir).

b) Desenvolva em *Python*, sobre a camada de abstracção acima desenvolvida e assegurando a independência da implementação, uma função `sucessos` que recebendo uma lista de elementos `w` e uma função `tol`, devolve a lista das posições em `w` que correspondem a elementos que não abandonarão a fila formada inserindo sucessivamente cada elemento `x` de `w` com tolerância dada por `tol(x)`.

Grupo III (4 valores)

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

Recorde o algoritmo de ordenação por inserção. Implemente funcionalmente em *Python* uma versão do algoritmo de ordenação por inserção em que cada elemento seja inserido ordenadamente por pesquisa binária (a pesquisa da posição onde o elemento é inserido deve reduzir para metade, em cada passo, o número de possibilidades).

Grupo IV (3 valores)

Considere o seguinte programa imperativo `PROG`.

```
c=0
i=0
while i!=len(w):
    if w[i]==0:
        c=c+1
    i=i+1
```

Demonstre que é válida a asserção

$$\{\texttt{True}\}\ \texttt{PROG}\ \{0 \leq \texttt{c} \leq \texttt{len(w)}\}.$$