Interpretação e Compilação de Linguagens de Programação

Primeiro Teste 2012-2013 — 18 de Abril de 2013

Notas: O teste pode ser realizado com consulta de materiais em suporte papel trazidos pelo aluno. O exame tem a duração de 1h30. Podem ser anexadas folhas extra ao teste se necessário.

Nome:_____ Número:_____

Q-1 **[5 val.]** Esta pergunta trata da definição da sintaxe abstracta de uma linguagem de programação. Considere a linguagem *Acum* cuja sintaxe concreta é definida pela seguinte gramática:

$$\begin{array}{rcl} E & ::= & num \mid E_1{+}E_2 \mid x \mid \mathsf{decl}\ x = E_1\ \mathsf{in}\ E_2 \\ & \mid & \{E_1;\ldots;E_n\} \mid E_1\texttt{@}E_2 \\ & \mid & \mathsf{iter}\ x\ \mathsf{in}\ E_1\ \mathsf{acum}\ y = E_2\ \mathsf{with}\ E_3 \end{array}$$

As construções básicas da linguagem *Acum* são **os números inteiros** (num) e as operações habituais sobre os números inteiros, aqui simbolizadas na operação $E + E$; os **identificadores** (x); a **declaração de identificadores** $\mathsf{decl}\ x = E_1\ \mathsf{in}\ E_2$; as **coleções** definidas por enumeração $\{E_1;\ldots;E_n\}$, ou por concatenação $E_1\texttt{@}E_2$. A linguagem *Acum* também inclui a operação básica de iteração sobre coleções $\mathsf{iter}\ x\ \mathsf{in}\ E_1\ \mathsf{acum}\ y = E_2\ \mathsf{with}\ E_3$. Esta expressão itera a coleção denotada por E_1. O valor final da expressão é o valor da expressão E_3 para o último elemento da coleção. Na iteração i da expressão E_3, o identificador x denota o i-ésimo elemento da coleção, e o identificador y denota o valor da expressão E_3 na iteração $i-1$. O valor de y para $i = 0$ é dado pela expressão E_2. A linguagem *Acum* é avaliada segundo uma **estratégia de avaliação por valor** (*call-by-value*) e resolução estática de nomes.

Considere o seguinte exemplo escrito na linguagem *Acum*:

```
decl c = {1;2;3;4} in
  decl d = iter x in c acum y = {} with y@{x+1} in
    iter x in d acum y = 0 with x+y
```

a) **[3 val.]** **Descreva** a sintaxe abstrata da linguagem *Acum* usando classes Java (nomes das classes e listas de campos), ou um tipo de dados abstrato (por exemplo, em ML ou Haskell).

Número:_____

b) [**2 val.**] **Represente** os valores da linguagem *Acum* usando classes Java (nomes das classes e listas de campos), ou um tipo de dados abstrato (por exemplo, em ML ou Haskell).

Q-2 [**8 val.**] Esta pergunta trata da definição da semântica de uma linguagem de programação.

a) [**6 val.**] **Defina** a semântica para as construções da linguagem *Acum* através da implementação do método **eval** de cada uma das classes correspondentes, uma classe visitante ou através de uma função de interpretação recursiva.

Número:_____

(continuação da alínea 1.c))

b) [**1 val.**] **Indique** o resultado da expressão em exemplo na questão **Q-1**.

c) [**1 val.**] **Enumere** os erros de execução que podem ocorrer durante a execução de um programa escrito na linguagem *Acum*.

Número:_____

Q-3 **[7 val.]** Esta questão é sobre aspetos da semântica de programas e estratégias de avaliação. Na linguagem *Acum*, extendida com closures (construções fun $x \to E$ e $E_1 E_2$) considere o exemplo

```
decl c = {1;2;3;4} in
  decl g = fun f → iter x in c acum y = {} with y@{f x} in
    decl x = 10 in decl h = fun y → x+y in
      g h
```

a) **[2 val.]** **Indique** os passos de redução e qual o resultado de avaliar a expressão com resolução estática de nomes.

b) **[3 val.]** **Indique** os passos de redução e qual o resultado de avaliar a expressão com resolução dinâmica de nomes.

Q-4 **[2 val.]** "A linguagem C não implementa fechos mas no entanto é possível passar funções por parâmetro e usar funções como resultado de outras funções". **Justifique** a frase anterior e **compare** com a linguagem Pascal onde as funções só podem ser passadas para um contexto interno e não usadas como resultados de outras funções.