Exame de Linguagens de Programação 2

26 de Junho de 2001 (Duração: 2H30M)

I

1. [3] Indique o valor dos programas PCF seguintes:

   (a)
   ```
   let y = 1
       in let f = (fun x -> y)
         in let y = 0 in f(0)
   ```

   (b)
   ```
   let x = 0 in
       letrec f = fun x -> if(x, x+f(x-1), 0) in f(5)
   ```

   (c)
   ```
   let f = fun x,y -> x(y) in
       let g = fun y -> fun x -> f(x,y) in
         in  r = g(0) in r(fun y -> y+1)
   ```

2. [2] Considere o programa I(a):

   (a) Qual o valor associado no ambiente ao identificador `f` no momento da avaliação da expressão `f(0)`?

   (b) Qual seria o valor do programa se a linguagem PCF usasse a regra de resolução dinâmica de nomes? Justifique.

II

Considere a linguagem Oby estudada nas aulas. A linguagem Oby é muito poderosa apesar de muito simples, pois permite "programar" praticamente todas as construções das linguagens de programação modernas. Mostre como é possível definir um tipo de dados `Vector` em Oby, definindo uma função `Vector` que sempre que chamada com um número natural n devolve um *novo* vector de dimensão n, e funções `get`(vec, n), `set`(vec, n, val) e `size`(vec), que permitem respectivamente:

- Aceder ao valor guardado na posição n do vector *vec*.
- Alterar para *val* o valor na posição n do vector *vec*.
- Consultar a dimensão do vector *vec*.

Por exemplo, usando estas funções poderiamos escrever uma função para somar os elementos de um vector de inteiros assim:

```
let sumaux = fun v, n -> if (n) get(v,n)+sumaux(v,n-1) else 0
  in (fun v -> sumaux(v,size(v)));
```

Ou construir um vector com os primeiros três números naturais assim:

```
let v = Vector(3) in { set(v,1,1); set(v,2,2); set(v,3,3); v }        (B)
```

1. [2] Defina em Oby as funções `Vector`, `set`, `get` e `size`.

2. [1] Indique qual é o estado do ambiente e da memória do interpretador de Oby durante a avaliação do programa `(B)` acima, no momento em que é avaliado o último elemento (sublinhado) do bloco

   $$\cdots \{ \texttt{set(v,1,1); set(v,2,2); set(v,3,3);} \underline{\texttt{v}} \}$$

3. [2] Uma linguagem de programação diz-se induzir fenómenos de *aliasing* se é possível, em tempo de execução, referir uma mesma posição de memória usando expressões sintácticas diferentes.

   (a) A linguagem Oby pode induzir *aliasing*? Exemplifique em caso positivo.

   (b) Numa linguagem funcional pura nunca surgem fenómenos de *aliasing*. Porquê?

III

A linguagem Oby permite utilizar registos, criados com expressões da forma

$$[l_1 = v_1; \cdots; l_n = v_n]$$

onde os l_i são etiquetas (identificadores) e os v_i são os valores associados a cada etiqueta. Os "campos" de um registo são acessíveis através da operação de selecção usual (usando o ponto). Por exemplo, o valor do programa seguinte é 3.

```
let pack = [ a = 1; b = 2 ] in pack.a + pack.b;
```

1. [2] Considere a seguinte representação de uma classe de objectos contadores

```
let counterclass = ( fun ->
     let c = new 0 in
       [
         inc = fun -> c := !c+1;
         dec = fun -> c := !c-1
       ]
     ) in ...
```

   (a) Neste tipo de representação não é possível que um "método" use um outro método do mesmo objecto. Por exemplo, não seria possível definir o método

```
dup = fun -> { ... inc(); ... inc(); }
```

   Explique porquê.

   (b) Apresente uma solução para o problema descrito em (a), indicando (I) como escreveria a "classe" com o método `dup` e (II) como utilizaria a "classe" para criar novos objectos (Sugestão: considere que uma classe é representada como uma função que toma como argumento a referência para o objecto (*this*), e devolve o objecto construído.)

2. A linguagem Oby define também uma operação de concatenação `concat`$(-,-)$ para valores de tipo *string*. No entanto, a operação de concatenção também pode ser útil no caso dos registos. Em geral, o resultado de concatenar um registo r_1 com um registo r_2 é um novo registo, onde aparecem todas as etiquetas de r_1 e de r_2, e o valor associado a cada etiqueta é tal que se a etiqueta existir nos dois registos, o valor definido por r_1 tem prioridade. Por exemplo, o valor de

```
let r1 = [a=1;b=2]
  in let r2 = [b=3; c=4]
     in concat(r1,r2)
```

   é o registo `[a=1;b=2;c=4]`, enquanto o valor de

```
let r1 = [a=1;b=2]
  in let r2 = [b=3; c=4]
     in concat(r2,r1)
```

   é o registo `[a=1;b=3;c=4]`.

   (a) [1] Indique detalhadamente como alteraria o interpretador de Oby que desenvolveu nas aulas práticas de modo a que este suporte a concatenação de registos, com a semântica apresentada.

   (b) [2] Discuta como poderia utilizar a operação de concatenação de registos para representar em Oby a herança entre classes de objectos com a semântica usual (isto é, métodos definidos na classe derivada substituem os métodos com o mesmo nome na classe base).