Linguagens de Programação 2009/2010

Departamento de Informática, Universidade de Évora

2º Teste

25 de Junho de 2010

Observações:

- A prova tem a duração de **2 horas** e é **sem consulta**.
- Todas as respostas devem ser devidamente **justificadas**.

1. Considere o fragmento de código seguinte:

```
int x, y;
int f( int a, int b ){
   a=a*b; b=y; y=5-x; return b;
}
x=3; y=2;
x=f( y, y );
print x,y;
```

   Diga, **justificando**, quais os valores escritos pelo programa

   (a) quando a passagem de parâmetros é feita por valor?

   (b) quando a passagem de parâmetros é feita por referência?

2. De que modo a recursividade terminal pode ser utilizada na implementação de uma linguagem para melhorar a sua eficiência em termos de espaço alocado e velocidade de execução?

3. Considere o seguinte programa ML:

```
1    val x=4;
2    val y=2;
3    fun f x = let val y=4 in x+y end;
4    fun g h = let val x=3 in f y + (h x) end;
5    g f;
```

   (a) Para cada uma das ocorrências de `x` e `y` (linhas 3 e 4) indique, **justificando**, qual o tipo de variável (local, global, parâmetro, ...).

   (b) Qual o valor calculado pelo programa? Justifique mostrando o conteúdo da pilha de execução e descreva os passos efectuados que permitem obter o valor de cada variável.

   (c) Numa linguagem com ligação dinâmica de nomes qual seria o resultado? Justifique.

4. Numa linguagem com funções de nível superior, qual a dificuldade/complexidade acescida (face a uma linguagem sem funções de nível superior) que surge quando uma função devolve outra função como resultado?

5. Considere a seguinte função ML que utiliza a excepção `Impar`:

```
fun f( 0, count ) = count
  | f( 1, count ) = raise Impar
  | f( n, count ) = f( n-2, count+1 ) handle Impar => count;
```

Qual o resultado da avaliação da expressão `f(7,0)`? Justifique indicando os passos executados. Inclua apenas os seguintes tipos de passos:

- `call` *func(args)*: chamada de função *func* com argumentos *args*
- `return` *res* `de` *func(args)*: retorno de função *func(args)* com resultado *res*
- `raise` *excpt*: levantamento da excepção *excpt*
- `pop` *func(args)*: pop do registo de activação correspondente à chamada de função *func(args)*
- `handle` *excpt* `em` *func(args)*: tratamento da excepção *excpt* no corpo *func(args)*

6. Considere um tipo `Num` com dois subtipos, `Float` e `Int`, e as seguintes funções

```
Num f( Num )        Num g( Float )        Int h( Num )
```

Se o fragmento de código `f(g(x))` estiver bem tipado, indique, **justificando**, se se obtém uma expressão bem tipada substituindo esse fragmento por

(a) `g(f(x))`

(b) `f(f(x))`

(c) `h(h(x))`

(d) `g(h(x))`