Paradigmas de Linguagens de Programação
Exame Escrito
Centro de Informática – UFPE, 12 de junho de 2012

Questão 1 [2,0] Defina uma função ***f*** que recebe duas listas de pares de um mesmo tipo e retorna uma lista formada por todos os elementos da segunda lista, além dos elementos da primeira lista cujo primeiro elemento do par não coincide com qualquer primeiro elemento dos pares da segunda lista. Por exemplo:

f([(1,a),(2,b),(3,c)],[(2,d),(4,e)]) = [(1,a),(3,c),(2,d),(4,e)]

Duas observações:

a) A ordem dos pares não importa
b) O primeiro elemento de cada par não é repetido em uma mesma lista

Questão 2 [2,0] Defina e prove, por indução, uma propriedade que garante que a função ***f*** aplicada a 2 listas iguais resulta a própria lista.

Questão 3 [2,0] Assinale com V (para verdadeiro) ou F (para falso):

( ) Em qualquer que seja a computação, uma avaliação estrita é sempre mais eficiente do que uma avaliação sob demanda (*lazy evaluation*).

( ) O Princípio da Correspondência (*the Correspondence Principle*) sugere que, para cada forma de declaração, deve existir um mecanismo de passagem de parâmetros, e vice-versa.

( ) A propriedade de universalidade define que uma linguagem de programação deve ter o poder de expressão de uma máquina de Turing.

( ) O escopo de uma variável é a porção do texto do programa onde a variável pode ser utilizada; tal porção é sempre contínua (ininterrupta).

Questão 4 [3,0] Estenda a Linguagem Funcional 1 com a possibilidade de definir tipos enumerados, usando a seguinte sintaxe ***type t = e1 | e2 | ... | en***, de forma que tipos enumerados possam ser definidos em expressões ***let*** e utilizados da mesma forma que os tipos primitivos (**assuma que os valores dos tipos enumerados são disjuntos e diferentes de nomes de variáveis, funções e parâmetros**). Deve ser possível definir um numero arbitrário de tipos, funções e variáveis em uma mesma expressão ***let***. Por exemplo, o seguinte programa é válido e retorna ***"Aprovado"***:

```
let type Status = AM | AP | RP in
let var aluno1 = AP, fun resultado st =
                        if st == AM then "Aprovado por Media"
                        else if st == AP then  "Aprovado"
                             else  "Reprovado" in
resultado(aluno1)
```

Elabore uma implementação parcial desta extensão, considerando o seguinte:

1) Defina a BNF para a linguagem redefinida, destacando apenas o que mudar.
2) Explique o que precisaria mudar no *parser* e nos ambientes de compilação e execução, destacando que novos atributos e métodos seriam necessários nestes ambientes.
3) Indique que novas classes seriam necessárias e que classes existentes precisariam ser modificadas. Escolha uma das classes e implemente a nova versão do método ***avaliar()***.

Boa Sorte (**Adicione 1 ponto em uma questão de sua escolha**)

Apêndice 1. BNF de LF1.

Programa ::= Expressao

Expressao ::= Valor
| ExpUnaria
| ExpBinaria
| ExpDeclaracao
| Id
| Aplicacao
| IfThenElse

Valor ::= ValorConcreto

ValorConcreto ::= ValorInteiro | ValorBooleano | ValorString

ExpUnaria ::= "-" Expressao | "not" Expressao | "length" Expressao

ExpBinaria ::= Expressao "+" Expressao
| Expressao "-" Expressao
| Expressao "and" Expressao
| Expressao "or" Expressao
| Expressao "==" Expressao
| Expressao "++" Expressao

ExpDeclaracao ::= "let" DeclaracaoFuncional "in" Expressao

DeclaracaoFuncional ::= DecVariavel
| DecFuncao
| DeclaracaoFuncional "," DeclaracaoFuncional

DecVariavel ::= "var" Id "=" Expressao

DecFuncao ::= "fun" Id Id Id "=" Expressao

Aplicacao:= Id"("Expressao, Expressao")"

IfThenElse ::= "if" Expressao "then" Expressao "else" Expressao

Classes Auxiliares

ValorFuncao
AmbienteExecucaoFuncional
ContextoExecucaoFuncional