Paradigmas de Linguagens de Programação
Exame Escrito
Centro de Informática – UFPE, 14 de julho de 2016

Questão 1 [2,0] Defina as funções ***E*** e ***D*** sobre listas. A primeira retorna os primeiros elementos de uma lista até uma quantidade ***n*** informada; a segunda retorna do elemento ***n+1*** até o último elemento. Por exemplo, ***E([a,b,c,d],2)*** = ***[a,b]*** e ***D***(***[a,b,c,d],2)*** = ***[c,d]***. Pode assumir que a quantidade ***n*** de elementos é um número entre 0 e o tamanho da lista.

Questão 2 [2,0] Prove, por indução, para ***n*** e listas ***xs***, a seguinte propriedade:

E(xs,n) ++ D(xs,n) = xs

onde **++** é o operador de concatenação de listas, como visto em sala. **[Dicas: o caso base deve considerar uma lista vazia e *n = 0*; na prova do passo indutivo assumir que a propriedade vale para *xs* e *n* e provar para *x:xs* e *n+1*. Será necessário usar uma equação da definição de concatenação de listas]**

Questão 3 [1,0] Explique o conceito de *aliasing* em programação e a relação entre este conceito e passagem de parâmetro por referência.

Questão 4 [2,0] Assinale com V (para verdadeiro) ou F (para falso):

( ) Considere uma função ***listaInfinita(n)*** que gera uma lista infinita com elementos ***[n,n+1, ...]***. A expressão ***head(listaInfinita(n))*** termina e retorna o elemento ***n*** se o mecanismo de avaliação for estrito (*innermost*).

( ) Segundo o Princípio Qualificação, uma linguagem deve oferecer blocos de declarações para todas as estruturas sintáticas da linguagem que especifiquem alguma forma de computação.

Questão 5 [3,0] Estenda a Linguagem Imperativa 2 (veja BNF em anexo), modificando procedimentos para retornarem um valor (como métodos não *void* em Java); o tipo de retorno deve ser especificado. O corpo de um procedimento passa a ser um comando (como antes), seguido do novo comando ***return E***, onde ***E*** é uma expressão. A chamada de um procedimento deixa de ser um comando na linguagem e passa a ser uma expressão. A implementação deve considerar os métodos de avaliação e checa tipo, bem como as classes novas ou modificadas, impactadas por esta mudança. Particularmente:

1) Defina a BNF para a linguagem redefinida, destacando apenas o que mudar.
2) Explique se é necessária alguma mudança nos ambientes de compilação e execução.
3) Implemente novas classes, se for o caso, e indique todas as classes que seriam afetadas (ilustre a modificação em pelo menos uma classe impactada).

Boa Sorte

Apêndice 1. BNF de LI2.

Programa ::= Comando

Comando ::= Atribuicao | ComandoDeclaracao
| While | IfThenElse
| IO | Comando ";" Comando
| Skip | ChamadaProcedimento
Atribuicao ::= Id ":=" Expressao
Expressao ::= Valor | ExpUnaria | ExpBinaria | Id
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

ComandoDeclaracao :: = "{" Declaracao ";" Comando "}"

Declaracao ::= DeclaracaoVariavel | DeclaracaoProcedimento
| DeclaracaoComposta
DeclaracaoVariavel ::= "var" Id "=" Expressao
DeclaracaoComposta ::= Declaracao "," Declaracao
DeclaracaoProcedimento ::= "proc" Id "(" [ListaDeclaracaoParametro] ")" "{" Comando "}"
ListaDeclaracaoParametro ::= Tipo Id | Tipo Id "," ListaDeclaracaoParametro
Tipo ::= "string" | "int" | "boolean"
While ::= "while" Expressao "do" Comando
IfThenElse ::= "if" Expressao "then" Comando "else" Comando
IO ::= "write" "(" Expressao ")" | "read" "(" Id ")"
ChamadaProcedimento ::= "call" Id "(" [ListaExpressao] ")"
ListaExpressao ::= Expressao | Expressao, ListaExpressao

Classes Auxiliares

AmbienteCompilacaoImperativa2

AmbienteExecucaoImperativa2
ContextoCompilacaoImperativa2
ContextoExecucaoImperativa2
ListaValor
Procedimento
DefProcedimento
ProcedimentoJaDeclaradoException
ProcedimentoNaoDeclaradoException