Paradigmas de Linguagens de Programação
Exame Escrito

Augusto Sampaio e Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

5 de julho de 2000

Questão 1 Defina uma função de alta ordem (na LF2) que recebe como parâmetros uma lista de funções e um inteiro, retornando como resultado o mesmo valor retornado pela primeira função na lista que quando aplicada ao argumento inteiro gera um valor maior que 100. Se não houver tal função na lista, a função definida deve retornar 0. Assuma que a LF2 tem o tipo lista e o operador relacional ">".

Questão 2 Descreva e compare, usando exemplos, os mecanismos de passagem de parâmetros por valor, referência e valor-resultado.

Questão 3 Implemente a função fatorial de forma imperativa (na LI1) e funcional (na LF1) e prove que uma das implementações garante que o fatorial de um número é sempre maior que o próprio número. Justifique a escolha da implementação que você usou para realizar a prova.

Questão 4 Considere a seguinte implementação do interpretador do comando `throw` de uma linguagem imperativa:

```
class Throw implements Comando {
  private Expressao expressao;
  Throw (Expressao e) {expressao = e;}
  AmbienteExecucao executar(AmbienteExecucao a) throws ThrowException {
     throw new ThrowException(expressao.avaliar(a),a);
  }
}
```

onde as exceções são representadas por valores (inteiros, booleanos ou strings) e a classe `ThrowException` contém dois atributos—um valor e um ambiente—inicializados pelo construtor e retornados pelos métodos `getValor` e `getAmbiente`. Seguindo o estilo desta implementação, implemente um interpretador para o comando `try-catch-finally` com funcionalidade similar ao de Java mas onde a escolha de um `catch` é feita com base no valor da exceção, não no seu tipo. Lembre-se que para um determinado `try` pode haver zero ou vários `catch`, e que o `finally` é opcional desde que haja pelo menos um `catch`. Por simplicidade, considere que a linguagem em questão não permite a declaração de variáveis locais.