Segunda Prova de Linguagens de Programação
- DCC024 -
Ciência da Computação

Nome: _____

"Eu dou minha palavra de honra que não trapacearei neste exame."

Número de matrícula: _____

As regras do jogo:

- A prova é sem consulta.
- Quando terminar, não entregue nada além do caderno de provas para o instrutor.
- Quando escrever código, a sintaxe correta é importante.
- Cada estudante tem direito a fazer uma pergunta ao instrutor durante a prova. Traga o caderno de provas quando vier à mesa do instrutor[1].
- A prova termina uma hora e quarenta minutos após seu início.
- Seja honesto e lembre-se: **você deu sua palavra de honra**.

Alguns conselhos:

- Escreva sempre algo nas questões, a fim de ganhar algum crédito parcial.
- Se não entender a questão, e já tiver gasto sua pergunta, escreva a sua interpretação da questão junto à resposta.
- A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Questão 4	Q. Extra

[1]Se você perguntar ao instrutor qual a cor dos domingos, terá gasto a sua pergunta. O mesmo vale para "que horas são agora?", ou "posso usar meu direito de fazer uma pergunta?" Quando faltarem vinte minutos para o fim da prova, o instrutor dará o aviso.

1. Nesta questão você deverá implementar um predicado `multset` em Prolog. Para tanto, você irá implementar dois predicados auxiliares.

   (a) (3 Pontos) Implemente um predicado `rangelist(N, S)`, que seja verdade quando `S` for uma lista de tamanho entre 1 e `N` (inclusive). Por exemplo:

   ```
   ?- rangelist(2, [a, b]).
   true.

   ?- rangelist(3, [a, b]).
   true.

   ?- rangelist(1, [a, b]).
   false.

   ?- rangelist(2, L).
   L = [].
   L = [_G01].
   L = [_G01, _G02].
   ```

   (b) (3 Pontos) Escreva um predicado `submulset(S, L)`, que seja verdadeiro quando cada elemento de `S` estiver presente em L. Por exemplo:

   ```
   ?- submulset([a, b, a, b], [a, b]).
   true.

   ?- submulset([a, b, a, b], [a, b, c]).
   true.

   ?- submulset([a, b, c, a, b], [a, b]).
   false.
   ```

   (c) (4 Pontos) Escreva um predicado `multset(N, L, S)`, que seja verdadeiro quando os dois fatos abaixo forem verdade:

   - `S` for uma lista com tamanho entre 1 e `N` elementos (inclusive);
   - Cada elemento de `S` é um membro de `L`.

   Você pode usar os dois predicados feitos nas questões anteriores para resolver esse exercício. Caso não os tenha feito, você pode assumir a existência deles.

2. Esta questão refere-se aos dois programas abaixo, que estão escritos na linguagem de programação Kotlin. Você precisa tentar inferir, seguindo a sua intuição, qual o significado da sintaxe usada nos códigos abaixo para responder corretamente as perguntas que se seguem.

```
interface TreeNode {
  fun height(): Int
}

class TreeLeaf: TreeNode {
  override fun height(): Int {
    return 0
  }
}

class TreeValue(val v: Int, val l: TreeNode, val  r: TreeNode): TreeNode {
  override fun height(): Int {
    val lHeight = l.height()
    val rHeight = r.height()
    return 1 + if (lHeight > rHeight) lHeight else rHeight
  }
}

fun main(args: Array<String>) {
  val numbers = listOf(2, 5, 3, 8, 1, 29, 43, 2, 15, 6, 1)
  val tree: TreeNode = numbers2Tree(numbers)
  println("Obj Height = " + tree.height())
}
```

```
interface TreeNode {
}

class TreeLeaf: TreeNode {
}

class TreeValue(val v: Int, val l: TreeNode, val  r: TreeNode): TreeNode {
}

fun height(treeNode: TreeNode): Int {
  return when (treeNode) {
    is TreeLeaf -> 0
    is TreeValue -> 1 + max(height(treeNode.l), height(treeNode.r))
    else -> throw IllegalArgumentException("Unknown type")
  }
}

fun main(args: Array<String>) {
  val numbers = listOf(2, 5, 3, 8, 1, 29, 43, 2, 15, 6, 1)
  val tree: TreeNode = numbers2Tree(numbers)
  println("Obj Height = " + height(tree))
}
```

(a) (1 Ponto) Um desses programas implementa a função `height` de forma claramente orientada a objetos. O outro, implementa a mesma função de forma claramente procedural. Qual desses códigos é o mais orientado a objetos. Responda com "código à esquerda"ou "código à direita".

(b) (3 Pontos) Caso fosse necessário adicionar um novo tipo de nó à árvore, por exemplo, um nó ternário, qual das versões acima seria mais fácil de alterar? Justifique a sua resposta.

(c) (3 Pontos) Caso fosse necessário adicionar uma nova operação aos programas acima, por exemplo, para imprimir o conteúdo de uma árvore, qual dos paradigmas de programação (procedural ou orientado a objetos) acima seria mais fácil de ser estendido? Novamente, é necessário que você justifique a sua resposta.

(d) (3 Pontos) É possível transformar uma lista de números em uma árvore usando o código abaixo:

```
val numbers = listOf(2, 5, 3, 8, 1, 29, 43, 2, 15, 6, 1)
val initTree: TreeNode = TreeLeaf()
val tree: TreeNode = numbers.fold(initTree) { acc, i -> insert(acc, i) }
```

Explique o que é a sintaxe marcada em cinza. Mesmo que você jamais tenha visto Kotlin, é esperado que você possa relacionar tal sintaxe a conceitos já vistos no curso de linguagens de programação. Para ajudar na resposta, note que a função `insert` recebe uma árvore T e um número N e produz uma nova árvore $T' = T \cup \{N\}$.

3. Esta questão refere-se à forma como programas são executados em diferentes linguagens de programação.

   (a) (3 Pontos) Programas escritos em `bash` script são puramente interpretados. Isto é, a árvore de sintaxe deles é interpretada, e código nunca é gerado para alguma máquina alvo. Por que `bash` é normalmente executada desta forma?

   (b) (3 Pontos) Programas escritos em C normalmente são compilados. Por que C é uma linguagem normalmente compilada?

   (c) (4 Pontos) Programas escritos em Java são tipicamente virtualizados. Isto é, código fonte é compilado para uma linguagem de *bytecodes*, e esta linguagem é então interpretada pela máquina virtual Java. Porque java é executada dessa forma, em vez de ser compilada como em C, ou em vez de ser puramente interpretada, como em `bash`?

4. John Backus, quando recebeu o Prêmio Turing, em 1978, fez um discurso que continha o seguinte trecho:

> My point is this: while it was perhaps natural and inevitable that languages like FORTRAN and its successors should have developed out of the concept of the von Neumann computer as they did, the fact that such languages have dominated our thinking for twenty years is unfortunate. It is unfortunate because their long-standing familiarity will make it hard for us to understand and adopt new programming styles which one day will offer far greater intellectual and computational power.

(a) John Backus ganhou o Prêmio Turing por ter inventado a linguagem de programação FORTRAN. FORTRAN, sendo uma das primeiras linguagens de programação criada, é considerada membro da *Primeira Geração de Linguagens de Programação*. Cite três outras linguagens que fazem parte, junto com FORTRAN, da primeira geração de linguagens de programação, tendo sido criadas ainda nos anos 50, e que tiveram mais de 1000 usuários ao longo de sua história:

- (1 Ponto)
- (1 Ponto)
- (1 Ponto)

(b) (1 Ponto) FORTRAN é uma linguagem que pertence a qual paradigma de programação?

(c) (2 Pontos) Quais outros dois grandes paradigmas de programação existem, além daquele ao qual pertence FORTRAN?

- (1 Ponto)
- (1 Ponto)

(d) (2 Pontos) visto o discurso de John Backus, é fácil imaginar que o paradigma ao qual pertence FORTRAN possui algumas desvantagens. Cite uma desvantagem desse paradigma, quando comparado com os outros dois paradigmas da questão anterior.

5. Questão Extra:

   Qual o nome dos cinco Cavaleiros de Bronze que protagonizaram o *Anime* “Cavaleiros do Zodíaco”?

Se você está em dúvida sobre a resposta desta questão, e sente-se tentado a olhar a prova de seu colega ao lado, lembre-se que você deu a sua palavra de honra que não trapaceará neste exame [2].

[2] E lembre-se que a palavra de um verdadeiro cavaleiro não é fumaça