Exame Especial de Linguagens de Programação
- DCC024 -

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.
- A prova termina uma hora e quarenta minutos após seu início.

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.
- Serão avaliadas somente as sete melhores respostas. Então sinta-se livre para abandonar alguma questão devido ao tempo.
- A prova não é difícil, ela é divertida, então aproveite!

1. O objetive desta questão é desenvolver um sistema para representação de regiões geométricas. Uma região é um conjunto de pontos, e a única operação relevante no nosso sistema é pertinência: dado um ponto e uma região, queremos saber se o ponto está dentro da região. Representaremos uma região por meio de sua função característica, isto é, uma região é uma função que, dado um ponto, retorna verdadeiro se o ponto pertencer àquela região. Por exemplo, a função SML abaixo representa uma região circular com centro $(1.0, 3.0)$ e raio 4.5:

```
- fun circle (x,y) = (x - 1.0) * (x - 1.0) + (y - 3.0) * (y - 3.0) <= (4.5 * 4.5);
val circle = fn : real * real -> bool

- circle (1.1, 2.87);
val it = true : bool

- circle (10.0, 0.0);
val it = false : bool
```

   (a) (4 pontos) Sejamos mais abstratos que o exemplo dado: crie uma função `mc` em SML que, dado um centro (x, y) mais um raio r retorne uma região circular com as coordenadas dadas. A sua função deve ter a seguinte assinatura: `mc = fn : int * int * int -> int * int -> bool`.

   (b) (3 pontos) O interessante desta abordagem funcional é que podemos combinar regiões. Escreva uma função SML que, dada uma região g, retorna uma região g' que é o complemento de g. Uma região g' é o complemento de g se, e somente se, $g'(x, y)$ é verdade quando $g(x, y)$ é falso, e vice-versa. A sua função deve ter a seguinte assinatura: `complement = fn : (real * real -> bool) -> real * real -> bool`.

   (c) (3 pontos) Por fim, escreva a função SML `translate(g, dx, dy)` que faça a translação de uma região. Isto é, sendo g' a função retornada por `translate(g, dx, dy)`, então $g'(x, y)$ é verdade se, e somente se, $g(x + d_x, y + d_y)$ for verdade. A sua função deve ter a seguinte assinatura: `translate = fn : (real * real -> bool) * real * real -> real * real -> bool`.

2. (10 pontos) Lua é uma linguagem dinamicamente tipada, muito utilizada no desenvolvimento de videogames. Nesta linguagem, testes condicionais sobre as expressões `nil` e `false` retornam negativo. Testes condicionais sobre qualquer outra expressão retornam positivo. Os operadores `and` e `or` possuem a propriedade de *curto-circuito*. Entretanto, este operadores podem não retornar um valor booleano. Por exemplo, `"a" and "b" --> "b"`, pois `"a"` é considerado verdade, e o resultado da expressão é o valor da segunda sub-expressão. Considerando que o operador `and` possui maior precedência que o operador `or`, explique, clara e sucintamente, o que faz a função `enigma` abaixo:

```
function enigma(x, y) = x >= y and x or y
```

Ilustre a sua explicação com um pequeno conjunto de exemplos.

3. (10 pontos) Considere as duas classes abaixo, implementadas na linguagem Java:

```
class Mammal extends Animal {
  public void eat() {
    System.out.println("Mammal is eating");
  }
}

public class Animal {
  public void eat() {
    System.out.println("Animal is eating");
  }

  public static void main(String args[]) {
    Animal a = new Animal();
    Animal m = new Mammal();

    // Chamada 1:
    a.eat();

    // Chamada 2:
    m.eat();
  }
}
```

Existe alguma diferença, ainda que mínima, em termos de eficiência, entre a primeira chamada de método (`Chamada 1`) e a segunda (`Chamada 2`)? Em caso afirmativo, diga qual chamada é mais eficiente, e explique o por quê. Em caso negativo, justifique a sua resposta.

4. Existem muitos mecanismos de coleta de lixo diferentes; alguns destes mecanismos servem domínios bem específicos de aplicações. Abaixo são dados dois diferentes cenários. Para cada um, descreva um mecanismo de coleta de lixo que seria interessante para ele. Neste curso falamos dos seguintes algoritmos: contagem de referências, marcação e varredura, cópia e coleta. Além destes, você pode pensar em outros, mais adequados à aplicação em questão.

   (a) (5 pontos) Um sistema de tempo real, para controlar o braço de um robô que opera em uma linha de montagem. O principal requisito deste sistema é que ele deve responder a eventos em um certo período de tempo. Em hipótese alguma o sistema deve demorar mais que esta quantidade de tempo para produzir uma resposta.

   (b) (5 pontos) Um servidor web. O servidor recebe milhares de requisições, todas elas independentes umas das outras. Uma característica interessante deste sistema é que cada requisição tem um período de vida curto, e leva à criação de uma quantidade pequena, e muitas vezes previsível de dados.

5. (10 pontos) Escreva um predicado `lotto`, usando a linguagem Prolog, tal que `lotto(N, L, U, R)` receba um número inteiro `N`, um inteiro `L`, um inteiro `U` e seja verdade se `R` for uma lista com `N` elementos quaisquer da lista `[L, L+1, L+2, ..., U]`. Por exemplo:

```
?- lotto(2, 10, 15, L).
L = [10, 11] ;
L = [10, 12] ;
L = [10, 13] ;
L = [10, 14] ;
L = [10, 15] ;
L = [11, 12] ;
L = [11, 13] ;
L = [11, 14] ;
L = [11, 15] ;
L = [12, 13] ;
L = [12, 14] ;
L = [12, 15] ;
L = [13, 14] ;
L = [13, 15] ;
L = [14, 15] ;
false.
```

6. (10 pontos) Muitas linguagens de programação são compiladas diretamente para código de máquina. Outras são interpretadas. E outras executam sobre uma máquina virtual. É possível um programa executando sobre uma máquina virtual ser mais eficiente que um programa que foi compilado diretamente para código de máquina? Justifique a sua resposta.

7. Considere o programa abaixo, escrito em C++:

```
#include <stdio.h>
#include <string.h>
class Pencil {
  public:
    int p;
};
class Battleship {
  public:
    int p;
};
int main () {
  Pencil *p = new Pencil();
  p->p = 11;
  Battleship *b = (Battleship*)p;
  printf("p->p = %d, b->b = %d\n", p->p, b->p);
}
```

(a) (5 pontos) Qual o resultado da execução deste programa? Os resultados possíveis são:

- O programa não será compilado.
- O programa causará um erro em tempo de execução.
- O programa irá executar, e será impresso `p->p = 11, b->b = 11`.

(b) (5 pontos) De acordo com a sua resposta para a questão anterior, o que é verdade sobre a linguagem C++ (mais de uma resposta pode estar correta)?

- C++ é uma linguagem que verifica se as operações de coerção estão corretas estaticamente.
- C++ executa testes, em tempo de execução, para garantir que as coerções são utilizadas corretamente. Em caso de uso incorreto de coerções, C++ dispara uma exceção, como por exemplo `ClassCastException`.
- C++ não verifica se as coerções estão sendo usadas corretamente em nenhum momento.

8. (10 pontos) Duas formas de polimorfismo são a coerção e a sobrecarga. Na coerção implícita, um tipo é convertido em outro, sem a intervenção do programador. Por exemplo, em Java, durante a avaliação da expressão `1.0 + 2`, o segundo parâmetro é convertido de inteiro para número de ponto flutuante, e o resultado é dado em ponto flutuante. Na sobrecarga, temos o mesmo símbolo, ou função, representando diferentes operações. A combinação de coerção com sobrecarga pode complicar bastante o código, deixando-o próximo de ilegível. Algumas iterações são, na verdade, até proibidas pelo compilador, pois são ambíguas. Escreva um programa em C++, ou em Java, que, embora correto sintaticamente, não poderia ser compilador, pois o compilador não teria como desambiguar uma chamada de função devido à combinação de coerção com polimorfismo de sobrecarga. Explique onde a ambiguidade aparece em seu programa.