Primeira 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.
- 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!

Tabela 1: Pontos acumulados

Questão 1	Questão 2	Questão 3	Questão 4	Questão 5	Questão 6	Questão 7

1. Vários tipos de dados são conjuntos finitos de elementos. Por exemplo, o conjunto dos números inteiros, em Java, possui 2^{32} elementos. O maior inteiro é `2147483647`, e o menor é -2147483648. O tamanho deste conjunto faz parte da especificação da linguagem Java. Outras linguagens, como C ou SML, não definem os limites do conjunto de inteiros. Estes limites dependem da especificação do *hardware* onde a linguagem é implementada.

   (a) (5 pontos) Escreva uma função `biggestInt()` em C que encontre o valor do maior elemento presente no tipo `int`. Sua solução não pode usar as constantes definidas em `limites.h`, ou em qualquer outro arquivo. Você precisa ser capaz de computar este inteiro. Se a sua função demorar mais que 0.1 segundos para terminar, você perde um ponto.

   (b) (5 pontos) Escreva uma função `biggestInt` em SML que retorne o valor do maior elemento to tipo `int`. Sinta-se livre para passar quantos parâmetros quiser para a função. Dica, a função abaixo retorna zero caso a soma de dois inteiros cause um derramamento:

   ```
   fun sum x y = x + y
       handle Overflow => 0;
   ```

   Novamente, se a sua solução demorar mais que 0.1 segundos para terminar, você perde um ponto.

2. Considere a função SML abaixo, que calcula o enésimo número da sequência de Fibonacci:

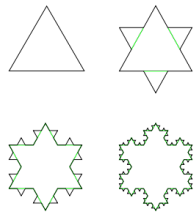
```
fun fib n = if n = 0 then 0 else if n = 1 then 1 else fib (n - 1) + fib (n - 2)
```

   (a) (1 ponto) Quantas vezes a função `fib` seria chamada para computar o quarto termo da sequência?

   (b) (2 pontos) Qual a complexidade assimptótica da função `fib`, em termos do valor de entrada n? Isto é, quantas vezes a função será recursivamente chamada para um dado valor de n?

   (c) (7 pontos) escreva uma função `fastFib`, cuja complexidade seja linear com base no valor da entrada n. Utilize tantas funções auxiliares quanto você julgar necessário.

3. (10 pontos) A *Curva de Koch* é um dos mais antigos e clássicos fractais já descritos na literatura. Exemplos de várias instâncias desta curva são dados na figura abaixo:



Alfabeto: F

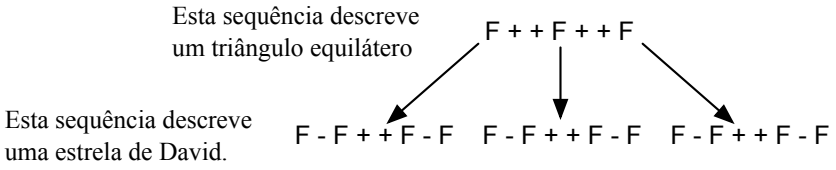
Constantes: +, -

Axioma: F++F++F

Produções: F ::= F-F++F-F

Interpretação:
F significa desenhe uma linha.
+ significa vire 60°
- significa vire 300°

A Curva de Koch é normalmente descrita via um sistema L, como este no centro da figura acima. Sistemas L são um formalismo baseado em gramáticas, desenvolvido pelo botânico *Aristid Lindenmayer* para descrever o crescimento de plantas. Ao contrário das gramáticas livres de contexto, que temos visto em aula, todas as produções são avaliadas ao mesmo tempo. As regras de um sistema L são aplicadas iterativamente, começando-se pelo axioma. A cada iteração, todas as regras possíveis são aplicadas. Caso as regras fossem aplicadas uma de cada vez, teríamos simplesmente uma linguagem, e não um sistema L. Assim, a figura abaixo descreve duas sequências válidas para a Curva de Koch, e uma inválida:



Porém, a sequência F-F++F-F++F-F++F-F++F não é válida, pois um dos F não foi expandido

Escreva uma gramática em Prolog que reconheça as três primeiras sequências válidas para a Curva de Koch. Dica, a primeira sequência possui três linhas, a segunda 12, e a terceira, 48. Outra dica, desta vez para relembrar a sintaxe, a gramática abaixo reconhece os números naturais:

```
number --> digit.
number --> digit, number.
digit --> [0] ; [1] ; [2] ; [3] ; [4] ; [5] ; [6] ; [7] ; [8] ; [9].
```

4. Considere o tipo algébrico `exp` abaixo, que descreve uma linguagem muito simples, com as operações aritméticas de adição, multiplicação e menos unário:

```
datatype exp = NUM of int |
               SUM of exp * exp |
               MUL of exp * exp |
               UNMINUS of exp
```

Implemente uma função `interpret`, de tipo `fn : exp -> int` que interprete esta linguagem. Seu interpretador deve assumir a semântica tradicional das operações aritméticas presentes. Por exemplo, data a expressão `val x = SUM((MUL (NUM 2, NUM 3)), SUM(UNMINUS (NUM 3), NUM 5))`, que corresponde à expressão aritmética $(2 \times 3) + ((-3) + 5)$, a chamada `interpret x` deve retornar o valor inteiro 8.

5. (5 pontos cada) Linguagens de programação provêem aos desenvolvedores diversas abstrações que têm o propósito específico de permitir reúso de código. As questões abaixo referem-se a duas destas abstrações:

   (a) Escreva um programa em SML que ilustre como funções de alta ordem são utilizadas como um mecanismo de reúso. Explique o que está sendo reusado neste exemplo.

   (b) Escreva um programa em SML que mostre como o polimorfismo paramétrico possibilita o reúso de código. Assim como na questão anterior, descreva o que está sendo reusado.

6. (2.5 pontos cada) Uma linguagem é estaticamente tipada quando o tipo de cada expressão pode ser resolvido em tempo de compilação. Uma linguagem é dinamicamente tipada quando o tipo da variável é resolvido em tempo de execução.

   (a) Dê um exemplo de uma linguagem estaticamente tipada. Como o compilador consegue descobrir o tipo das variáveis, no caso desta linguagem?

   (b) Dê um exemplo de uma linguagem dinamicamente tipada. Escreva um programa, muito simples, que evidencie o caráter dinâmico desta linguagem.

   (c) Cite uma vantagem da tipagem estática sobre a tipagem dinâmica.

   (d) Agora, cite uma vantagem da tipagem dinâmica sobre a tipagem estática.

7. Nós podemos representar números inteiros usando o cálculo λ. Uma das convenções mais comuns é assumir que um número n é uma função que recebe dois argumentos, e aplica o primeiro ao segundo n vezes. Por exemplo:

   - $0 = \lambda s.\lambda z.z$
   - $1 = \lambda s.\lambda z.sz$
   - $2 = \lambda s.\lambda z.s(sz)$

   Podemos também representar valores booleanos usando o cálculo λ. Por exemplo:

   - $F = \lambda x.\lambda y.y$
   - $T = \lambda x.\lambda y.x$

   (a) (3 pontos) Considere a função MUL $= \lambda n_1.\lambda n_2.\lambda z.n_1(n_2\ z)$. Usando a definição do número 2 acima, mostre todos os passos da redução MUL 2 2.

   (b) (3 pontos) Usando a função sucessor, SUCC $= \lambda n.\lambda y.\lambda x.y(n\ y\ x)$, defina a função ADD, que soma dois números.

   (c) (4 pontos) Defina uma função XOR, que receba dois valores booleanos b_1 e b_2, definidos como convencionado acima, e retorne T caso $b_1 \neq b_2$ e F caso contrário.