

Primeira Prova de Linguagens de Programação
- DCC024B -
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.
- Seja honesto e lembre-se: **você deu sua palavra de honra.**

Alguns conselhos:

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

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Extra

Questão extra (0.5): A edição 73 da revista *Rolling Stone Brasil* trouxe uma lista com as 100 maiores vozes da música brasileira. Cite um/a cantor/a que aparece entre os três primeiros nomes da lista.

1. Esta questão diz respeito à complexidade computacional da função abaixo, implementada em SML/NJ:

```
fun max [e] = e
  | max (h::t) = if max t > h then max t else h
```

- (a) (3 Pontos) A compilação da função `max` produz o seguinte aviso: `Warning: match nonexhaustive`. Por que?

- (b) (1 Pontos) Considere as duas chamadas de `max` abaixo. Qual delas terminaria mais rapidamente, a primeira chamada, ou a segunda?

```
- max [9, 8, 7, 6, 5, 4, 3, 2, 1] (* Primeira chamada *)
```

- max [1, 2, 3, 4, 5, 6, 7, 8, 9] (* Segunda chamada *)

- (c) (2 Pontos) Explique sua resposta para a questão (b) acima. Procure indicar quantas chamadas da função `max` ocorrem em cada caso. Caso seja difícil encontrar o número exato, indique a quantidade de chamadas em função de N , o número de elementos da lista.

- (d) (4 Pontos) Ré-escreva a função `max`, de modo que ela possua complexidade linear no número de elementos da lista de entrada, independente do conteúdo da lista. Note que tanto o melhor quanto o pior caso de complexidade da função implementada deve ser $O(N)$, sendo N o número de elementos da lista de entrada.

A large rectangular area filled with a uniform grid of small squares, resembling standard graph paper. The grid consists of 6 columns and 8 rows, providing ample space for drawing or writing.

Escreva sua resposta nesta grade, um caracter por célula. Não use mais células.

2. Esta questão refere-se à gramática abaixo, que representa uma linguagem muito simples de expressões booleanas:

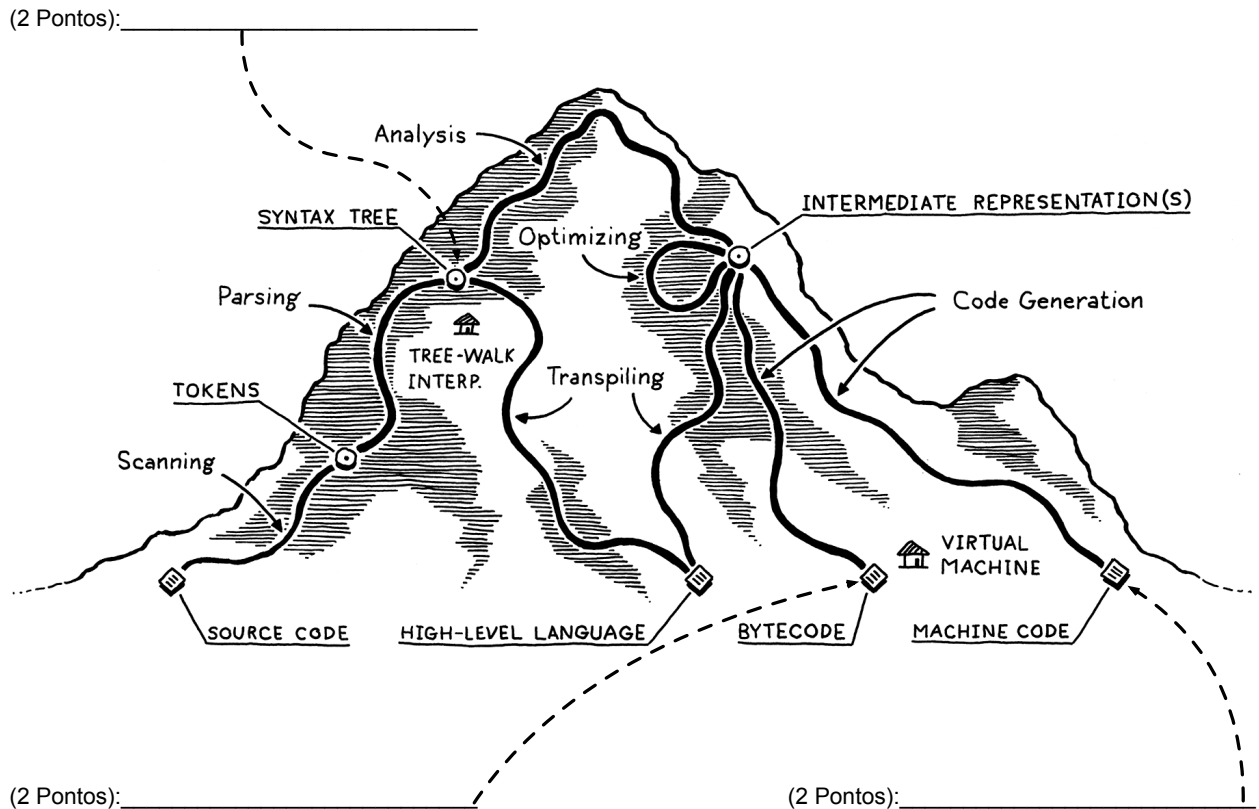
$$\begin{array}{lcl} \langle E \rangle & ::= & \langle E \rangle \text{ and } \langle E \rangle \\ & | & \langle E \rangle \text{ or } \langle E \rangle \\ & | & \text{not } \langle E \rangle \\ & | & \text{true} \\ & | & \text{false} \end{array}$$

- (a) (4 Pontos) A gramática acima é ambígua. Demonstre esse fato.
- (b) (2 Pontos) O que é a “*precedência relativa*” entre os operadores de uma gramática?
- (c) (1 Ponto) Quais operadores têm precedência maior: aqueles gerados por produções mais próximas do símbolo de partida da gramática, ou aqueles gerados por produções mais distantes? Note que a “*distância*”, neste caso, é o número de regras de produção expandidas até que um símbolo seja gerado.
- (d) (3 Pontos) Modifique a gramática, para que a operação **and** tenha precedência maior que a operação **or**. Não é necessário modificar a precedência dos outros termos que aparecem na linguagem.

3. Existem várias formas de executar um programa:

- Compilá-lo para linguagem de máquina.
- Interpretar sua árvore de sintaxe abstrata.
- Compilar o programa para *bytecodes* e interpretar os *bytecodes*.

Essas formas de execução são determinadas, em última instância, por qual representação do programa é usada durante a sua execução. A figura abaixo, retirada do livro “*Crafting Interpreters*”, indica essas várias possibilidades de execução. Indique uma linguagem de programação que é *majoritariamente* executa em algum dos caminhos indicados na figura.



(2 Pontos) Virtualmente toda linguagem de programação popular que é usada hoje possui o passo chamado “*parsing*”, visto na figura acima. Porque?

(2 Pontos) Algumas linguagens tendem a ser compiladas, enquanto outras tendem a ser interpretadas. Indique uma razão que faça com que uma linguagem seja normalmente compilada, em vez de ser interpretada.