Prova Final de Linguagens de Programação
- DCC024B -
Sistemas de Informação

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:

- 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	Questão 5	Questão 6

1. (10 Pontos) Nesta questão você deverá fazer um predicado, em Prolog, que determine se uma configuração do famoso "Jogo da Velha"é vitoriosa. Esse predicado deverá ser chamado `tictac`, e ele deverá receber uma lista `L` de nove elementos, contendo a configuração do jogo da velha que se quer analisar. Cada elemento de `L` pode ser um átomo `x`, ou um átomo `o`. A relação entre elementos de `L`, e posições no tabuleiro do jogo é dada pela figura abaixo:

`tictac([A, B, C, D, E, F, G, H, I]).`

A	B	C
D	E	F
G	H	I

Caso existam várias configurações vitoriosas, você pode optar por informar somente uma, ou por reportar todas elas. Neste caso, seu programa deverá ter a seguinte saída:

```
?- tictac([x, o, x, o, x, o, x, o, x]).
true ;
true.

?- tictac([x, o, x, o, x, x, o, o, x]).
true ;
true ;
false.

?- tictac([x, x, o, o, o, x, x, o, x]).
false.
```

2. A questão abaixo contém cinco itens. Cada um deles vale **2 Pontos**. Para responder cada item, considere a seguinte classe, implementada em Python:

```
class C:
  def __init__(self, name):
    self.name = name
  def __str__(self):
    return str(self.name)
```

A partir dessa definição acima, explique o que será impresso em cada item abaixo. Caso você ache que nada venha a ser impresso, devido a um erro em tempo de execução, escreva "Erro de execução", e explique porque o erro seria causado.

(a)
```
def foo0(p):
    p = C("Alvaro de Azevedo")
```
```
c = C("*")
foo0(c)
print "foo0: ", c
```

(b)
```
def foo1(p):
    p.name = "Bernard Shaw"
```
```
c = C("*")
foo1(c)
print "foo1: ", c
```

(c)
```
def foo2(p):
    p = "Clarice Lispector"
```
```
c = C("*")
foo2(c)
print "foo2: ", c
```

(d)
```
def foo3(p):
    p.newAttribute = "Daniel Defoe"
```
```
c = C("*")
foo3(c)
print "foo3: ", c.newAttribute
```

(e)
```
def foo4(p):
    p.name = 4
```
```
c = C("*")
foo4(c)
print "foo4: ", c
```

3. (10 Pontos) Escreva uma função `sqOdd`, em Python, que receba uma lista l de números, e retorne uma nova lista l' formada com o quadrado dos elementos ímpares de l. Veja os exemplos abaixo:

```
>>> sqOdd([2, 3, 5, 6, 7, 9])
[9, 25, 49, 81]

>>> sqOdd([])
[]

>>> sqOdd([2, 4, 6])
[]

>>> sqOdd([3, 5, 7, 9])
[9, 25, 49, 81]

>>> sqOdd([3, "a", 5])
TypeError: not all arguments converted during string formatting
```

4. A assinatura de um método em Java é data por uma combinação de três propriedades:

   (a) O nome do método.

   (b) O número de argumentos do método.

   (c) O tipo de cada argumento.

   Assim, é possível que tenhamos métodos de nomes iguais definidos em um mesmo espaço de nomes (o escopo de uma classe), desde que pelo menos uma das propriedades 2 e 3 seja suficiente para distinguir um método do outro.

   (a) (5 Pontos) Curiosamente, o tipo de retorno do método não é levado em consideração em sua assinatura. Mostre um exemplo de código, em Java, que seria ambíguo, caso tal não fosse o caso. Explique porque a ambiguidade existiria.

   (b) (5 Pontos) E quais propriedades são levadas em consideração na determinação da assinatura de métodos em Python? Explique porque as três propriedades enumeradas antes, para a linguagem Java, não poderiam ser aplicadas integralmente em Python.

5. A linguagem de programação Algol nunca se tornou muito popular entre programadores. Por outro lado, ela influenciou vários projetos subsequentes de linguagens de programação. Abaixo listam-se três influências de Algol. Em cada questão abaixo você deve: (i) explicar o quê aquele item significa. (ii) ilustrar sua explicação com um exemplo de código em uma linguagem conhecida e popular hoje. Mencione explicitamente qual linguagem você está usando.

   (a) (3 Pontos) Estrutura léxica de formato livre.

   (b) (2 Pontos) Procedimentos de primeira classe.

   (c) (3 Pontos) Escopo de blocos para variáveis locais.

   (d) (2 Pontos) Tipagem estática com anotações de tipos.

6. Uma função recursiva é dita de *cauda rasa* quando a última coisa que ela faz é chamar-se recursivamente. Re-implemente cada um dos predicados abaixo, para que eles sejam predicados de cauda rasa:

   (a) (3 Pontos) O predicado que soma os elementos de uma lista:

```
sum([], 0).
sum([H|T], X) :- sum(T, XAux), X is XAux + H.
```

   (b) (3 Pontos) O predicado que inverte os elementos de uma lista:

```
myappend([], L, L).
myappend([H|T], L, [H|LAux]) :- myappend(T, L, LAux).

myreverse([], []).
myreverse([H|T], R) :- myreverse(T, RT), myappend(RT, [H], R).
```

   (c) (4 Pontos) Afinal de contas, qual a vantagem de implementarmos funções de cauda rasa?