Prova Final 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. São perguntas: "Posso fazer uma pergunta?"ou "Quanto tempo falta?"
- 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	Extra

Questão Extra (0.5 Pontos): complete as próximas quatro palavras do trecho que falta da letra da seguinte música popular brasileira: "*Se avexe não. Observe quem vai subindo a ladeira. Seja princesa ou seja lavadeira. Pra ir mais alto,* _ _ _ _.".

1. (1 Ponto cada item) A maior parte das linguagens orientadas a objetos utiliza o conceito de classes para criar objetos. Por outro lado, é possível criá-los sem essa noção. A figura abaixo mostra duas maneiras de criar objetos sem que classes sejam necessárias:

Python

```
def createCounter(n):
    t = {'x': n}
    t['inc'] = lambda: t.update({'x': t['x'] + 1})
    t['get'] = lambda: t['x']
    return t

o0 = createCounter(1)
o0['inc']()
print o0['get']()

o1 = createCounter(100)
o1['inc']()
print o1['get']()
```

JavaScript

```
function createCounter(n) {
    function inc() {n++;}
    function get() { return n; }
    return {inc: inc, get: get};
}

var o0 = createCounter(1);
o0.inc();
alert(o0.get());

var o1 = createCounter(100);
o1.inc();
alert(o1.get());
```

(a) (1 Pontos) Que tipo de estrutura é retornada pela função `createCounter` no programa em Python?

(b) (2 Pontos) O estado dos objetos `o0` e `o1` no programa em Python pode ser alterado sem que o método `['inc']()` seja usado?[1] Justifique sua resposta.

(c) (1 Pontos) Que tipo de estrutura é retornada pela função `createCounter` no programa em JavaScript?

(d) (2 Pontos) O estado dos objetos `o0` e `o1` no programa em JavaScript pode ser alterado sem que o método `inc()` seja usado? Justifique sua resposta.

(e) (4 Pontos) Escreva uma classe cuja semântica seja similar à função `createCounter` no programa em Python. Instâncias de sua classe devem ter funcionalidades similares (não necessariamente iguais) àquelas dos objetos criados via `createCounter`. Não é necessário mostrar como objetos são instanciados.

[1] Nesta questão, e na letra (d) assuma que não existem outras referências apontando para a variável `n`.

2. Muitos autores definem Python como uma linguagem *multi-paradigmas*, o que significa que ela pode ser usada de forma imperativa ou declarativa. Nesta questão veremos algumas características funcionais dessa linguagem.

   (a) (3 Pontos) Mostre um exemplo de função de alta ordem sendo usada em Python. Você pode definir a função, ou pode reusar uma função que já existe.

   (b) (2 Pontos) Mostre um exemplo de *list comprehension* em Python.

   (c) (3 Pontos) Um programa escrito em Python parece um programa escrito em ML, no que toca o sistema de tipos, pois nessas duas linguagens quase sempre é possível escrever código sem especificar o tipo dos valores que existem no programa. Contudo, o sistema de tipos dessas duas linguagens é bem diferente. Qual é a principal diferença entre esses dois sistemas de tipos?

   (d) (2 Pontos) Mostre um exemplo de um programa que pode ser escrito em Python, mas cujo código equivalente, escrito em ML, não passaria pela verificação de tipos.

3. Esta questão se refere ao seguinte programa, escrito em uma pseudo-linguagem de programação:

```
function bar {
  int[] array = new int[2];
  array[0] = 0;
  array[1] = 2;
  foo(array[0], array[array[0]]);
  print(array[0], array[1]);
}

function foo (int x, int y) {
  x = 1;
  y = 10;
}
```

(a) Diga o que será impresso se o mecanismo de passagem de parâmetros nessa linguagem fosse cada um dos seguintes:

i. (2 Pontos) Passagem por valor.

ii. (2 Pontos) Passagem por referência.

iii. (2 Pontos) Passagem por expansão de macros.

(b) Um mecanismo de passagem de parâmetros pode ser estrito ou não-estrito (tardio). Classifique cada um dos mecanismos de passagem de parâmetros abaixo como estrito ou tardio:

- (1 Ponto) Passagem por valor.
- (1 Ponto) Passagem por referência.
- (1 Ponto) Passagem por expansão de macros.
- (1 Ponto) Passagem por necessidade.

4. Esta questão refere-se a três predicados em prolog: `car`, `pessoa` e `modelo`. Exemplos desses predicados podem ser vistos logo abaixo.

```
car(fiat, azul, joao).
car(fiat, verde, maria).
car(gol, prata, jose).
car(onyx, preto, joao).
car(up, preto, antonio).
car(onyx, prata, maria).
car(onyx, azul, telma).
```

```
pessoa(joao, professor, casado).
pessoa(maria, dentista, casado).
pessoa(jose, fazendeiro, solteiro).
pessoa(antonio, professor, casado).
pessoa(telma, policial, solteiro).
pessoa(lidia, fiscal, solteiro).
```

```
modelo(fiat, azul).
modelo(fiat, verde).
modelo(fiat, prata).
modelo(gol, preto).
modelo(gol, prata).
modelo(onyx, preto).
modelo(onyx, prata).
modelo(onyx, azul).
modelo(up, preto).
modelo(up, vermelho).
```

(a) (2 Pontos) Considerando os exemplos de predicados acima, qual o valor de `Carros` que torna o predicado abaixo verdadeiro?

```
findall(Carro, (modelo(Carro, Cor), not(car(Carro, Cor, _))), Carros).
```

(b) (2 Pontos) Complete o predicado abaixo. Ele é verdadeiro se `Pessoa` possui um carro azul:

```
possui_azul(Pessoa) :-
```

(c) (2 Pontos) Escreva um predicado `todas_cores(Pessoa, Cores)`, que seja verdadeiro caso `Cores` for a lista de todas as cores de carros que `Pessoa` possui. Por exemplo:

```
?- todas_cores(joao, Cores).
Cores = [azul, preto].
```

(d) (2 Pontos) Escreva um predicado `nao_tem_carro(Pessoa)`, que seja verdadeiro caso `Pessoa` nao possua nenhum carro. Por exemplo:

```
?- nao_tem_carro(Pessoa).
Pessoa = lidia.
```

(e) (2 Pontos) Escreva um predicado `solteiro_com_carro(Pessoa)`, que seja verdadeiro se `Pessoa` for uma pessoa solteira que possua pelo menos um carro. Por exemplo:

```
?- solteiro_com_carro(Pessoa).
Pessoa = jose ;
Pessoa = telma ;
```