Segunda 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 [1].
- 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	Q. Extra

[1] Se você perguntar ao instrutor qual a cor dos domingos, terá gasto a sua pergunta. O mesmo vale para "que horas são agora?", ou "posso usar meu direito de fazer uma pergunta?" Quando faltarem vinte minutos para o fim da prova, o instrutor dará o aviso.

1. Esta questão envolve a implementação de um predicado em Prolog, e sua análise de complexidade.

   (a) (4 Ponto) Implemente um predicado `aplast`, tal que `aplast(L, E, LL)` seja verdade quando a lista `LL` for a lista `L` com o elemento `E` concatenado ao seu final. Por exemplo:

   ```
   ?- aplast([a, b, c], d, X).
   X = [a, b, c, d].

   ?- aplast([a], d, X).
   X = [a, d].

   ?- aplast([], d, X).
   X = [d].
   ```

   (b) (3 Pontos) Qual a complexidade, em termos de tempo, do predicado `aplast`? Se você usar a notação O para descrever essa complexidade, explique em termos de quê o O é medido. Por exemplo: "logarítmica no tamanho da lista `L`".

   (c) (1 Ponto) qual será o valor de `LN` no programa abaixo? Note que estamos aplicando `length` sobre a lista `L`, não sobre a lista `X`.

   ```
   ?- L = [a, b, c], aplast(L, d, X), length(L, LN).
   ```

   (d) (2 Pontos) Estime o espaço (em termos de memória) usado por sua implementação de `aplast`. Sua resposta deve ser dada em função dos parâmetros de `aplast(L, E, X)`. Lembre-se que listas, em Prolog, são implementadas como cadeias de ponteiros, e que funções recursivas podem usar registros de ativação.

2. Esta questão refere-se à função abaixo, que cria "objetos"em Python:

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

Cada objeto possui um atributo `x`, e dois métodos: `inc` e `get`. O primeiro incrementa o valor de `x`, o segundo retorna este valor.

(a) Escreva um programa que realize as seguintes ações, na ordem dada:

i. (1 Ponto) crie um objeto com o atributo `x` igual a 10;

ii. (1 Ponto) incremente este valor;

iii. (1 Ponto) imprima o valor corrente de `x` usando a função `print` da biblioteca padrão Python.

iv. (1 Ponto) modifique a implementação de `get`, para que ela decremente, em vez de incrementar o valor de `x`.

(b) (4 Pontos) Nesta questão, você deverá simular herança usando a implementação de `createObj` como ponto de partida. Crie uma nova função `createObj2`, que produza objetos iguais àqueles construídos por `createObj`, exceto que o método `inc` desses novos objetos incrementa o valor de `x` de duas unidades, em vez de uma.

(c) (2 Pontos) Em Python é possível obter o tipo de um objeto criado como instância de uma classe em tempo de execução, como no exemplo abaixo:

```
>>> x = {1, 2, 3}
>>> if isinstance(x, set):
...   y = 2
... else:
...   y = "dois"
...
>>> print y
dois
```

Qual é o tipo dos objetos criados pelas funções `createObj` e `createObj2`?

3. Nesta questão você deverá comparar alguns aspectos de diferentes linguagens de programação.

   (a) (2 Pontos) Cite uma semelhança entre inteiros em C, e em Java.

   (b) (2 Pontos) Cite uma diferença entre inteiros em C, em em Java.

   (c) (2 Pontos) Cite uma diferença entre a forma como o compilador lida com exceções em Java em em SML.

   (d) (1 Ponto) Cite uma diferença entre a implementação de objetos em Java e em Python.

   (e) (1 Ponto) Cite uma diferença entre passagem de parâmetros em SML e em Haskell.

   (f) (1 Ponto) Cite uma diferença entre a implementação de polimorfismo paramétrico em Java e em C++.

   (g) (1 Ponto) Cite uma característica comum entre Python e Prolog, que não existe em ML[2].

[2]Não vale falar que Python e Prolog são linguagens que começam com a letra "P", ou têm seis letras – responda tecnicamente.

4. Esta questão refere-se aos predicados abaixo, escritos na linguagem Prolog[3]:

```
nomes([robertinho, melinha, dorinha, tanquin, mizinho, leleca]).
conhecido(X) :- nomes(N), member(X, N).
valido([X0, ama, X1]) :- conhecido(X0), conhecido(X1).
```

(a) (3 Pontos) Implemente um predicado `amados`, que produza uma lista de pares, contendo todos os casais em sentenças que obedecem a gramática abaixo, onde ϵ significa a sentença vazia:

$$\begin{array}{rcl} S & ::= & \epsilon \\ & | & \textit{nome } E \\ E & ::= & \textit{ama nome} \\ & | & \textit{ama nome que } E \end{array}$$

O termo *nome* denota um elemento N, tal que `conhecido(`N`)` seja verdade. Exemplos:

```
?- amados([], X).
X = []
?- amados([robertinho, ama, melinha], X).
X = [ (robertinho, melinha)]
?- amados([babete, ama, melinha], X).
false                                          # babete nao eh nome conhecido.
?- amados([robertinho, ama, melinha, que, ama, tanquin, que, ama, leleca], X).
X = [ (robertinho, melinha), (melinha, tanquin), (tanquin, leleca)]
```

(b) (3 Pontos) Implemente um predicado `visto(E, L)`, que seja verdade quando `E` for um nome conhecido que esteja presente na sentença `L`. Por exemplo:

```
?- visto(melinha, [robertinho, ama, melinha, que, ama, tanquin, que, ama, leleca]).
true.
?- visto(dorinha, [robertinho, ama, melinha, que, ama, tanquin, que, ama, leleca]).
false.
?- visto(tininha, [robertinho, ama, melinha, que, ama, tininha, que, ama, leleca]).
false.  # tininha nao eh um nome conhecido. Veja o predicado "nomes" acima.
```

(c) (4 Pontos) Implemente um predicado `casal(L, N1, N2)`, que seja verdade se a sentença `L` produzir os pares `(N1, N2)` e `(N2, N1)` via o predicado `amados`. Assuma que o predicado tenha sido implementado corretamente na parte (a) desta questão. Exemplos:

```
?- casal([melinha, ama, tanquin, que, ama, melinha], melinha, tanquin).
true ;
false.
?- casal([melinha, ama, tanquin, que, ama, dorinha], melinha, tanquin).
false.
```

[3] Você não precisa implementar o predicado `member`. O predicado `member(X, L)` recebe um elemento `X` e uma lista `L` e é verdadeiro se `X` pertence a lista `L`.

5. Questão Extra:

> "*It was so beautiful out on the country, it was summer- the wheat fields were golden, the oats were green, and down among the green meadows the hay was stacked. There the stork minced about on his red legs, clacking away in Egyptian, which was the language his mother had taught him. Round about the field and meadow lands rose vast forests, in which deep lakes lay hidden. Yes, it was indeed lovely out there in the country. ...*"

Assim começa *O Patinho Feio*, uma das estórias mais conhecidas entre os povos humanos que habitam o planeta terra. Para ganhar um ponto extra, você deve responder duas questões. Primeiro, em qual língua foi originalmente escrita a estória? Segundo, como termina a estória?

E se você pensa que "O Patinho Feio"não tem nada a ver com linguagens de programação, lembre-se do sistema de tipagem usado em Python, Php, JavaScript e muitas outras linguagens. E se você está em dúvida sobre a resposta desta questão, e sente-se tentado a olhar a prova de seu colega ao lado, lembre-se ainda que você deu a sua palavra de honra que não trapaceará neste exame [4].

[4] E lembre-se que a palavra de um verdadeiro cossaco não é fumaça