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.
- 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	Extra 0.5

Questão extra (0.5): cite alguma descoberta científica que é creditada a Galileu Galilei.

1. Para cada um dos itens abaixo, responda se a informação pertinente àquele item é encontrada em tempo de compilação ou em tempo de execução. Cada item vale dois pontos.

   (a) O tipo da variável `i`, na linha 2 do programa abaixo, escrito em C:

   ```
   1: int i = 3;
   2: float f = 0.1416F + i;
   ```

   (b) O valor de `x`, no programa abaixo, escrito em C:

   ```
   int* p = (int*) malloc(read() * sizeof(int));
   int x = sizeof(p);
   ```

   (c) A implementação do método `eat()` no programa abaixo, escrito em Java:

   ```
   Animal a = read() ? new Animal() : new Abelha();
   a.eat();
   ```

   (d) O tipo da função `makeFirst`, no programa abaixo, implementado em SML:

   ```
   fun first (a, _) = a;
   val x = true ;
   fun makeFirst t =  if x then 1 + first t else 1 - first t
   ```

   (e) A quantidade de métodos que podem ser invocados a partir do objeto `a`, no programa abaixo, escrito em Java:

   ```
   Animal a = new Animal();
   if (read() == '\0') {
     a = new Abelha();
   }
   ```

2. Esta questão refere-se ao programa abaixo:

```
int main() {
  int x = 1;
  int y = 2;
  int z = 3;
  int a = x << y << z;
  int a1 = ((x << y) << z);
  int a2 = (x << (y << z));
  int a3 = 1 << 2 << 3;
  int* p = &x;
  *p = 2;
  printf("%d\n", x);
  printf("%d, %d, %d, %d\n", a, a1, a2, a3);
}
```

(a) (4 Pontos) O programa acima imprime o seguinte resultado:

```
2
32, 32, 65536, 32
```

Dados esses resultados, qual é a associatividade da operação de arredamento para a esquerda?

(b) (4 Pontos) Que resultado seria impresso se a associatividade da operação de arredamento para a esquerda fosse invertida?

(c) (2 Pontos) Cite um operador de SML que possui associatividade contrária àquela respondida no item (a) desta questão.

3. Nesta questão você deverá implementar uma função `removeDups` em Python. Essa função recebe uma lista `L`, e remove de `L` todas as cópias de elementos duplicados. Por exemplo:

```
>>> L = []
>>> removeDups(L) ; L
[]
>>> L = [1, 1, 1, 1, 1]
>>> removeDups(L) ; L
[1]
>>> L = [2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1]
>>> removeDups(L) ; L
[2, 1]
```

(a) (3 Pontos) Implemente a função `removeDups` usando o espaço ao lado.

(b) (1 Pontos) Que sintaxe você usou para remover elementos de uma lista?

(c) (1 Ponto) você removeu elementos da lista enquanto estava iterando por ela?

(d) (3 Pontos) Qual é a complexidade asimptótica do código que você implementou na questão (a)?

(e) (2 Pontos) Seria possível reduzir essa complexidade? Explique como você faria isso (não é necessário código), ou exponha um argumento que mostre que tal não é possível.

4. Esta questão refere-se aos predicados abaixo, que implementam um grafo:

```
edge(a, b, 3).
edge(a, c, 5).
edge(b, c, 2).
edge(c, a, 8).
edge(c, d, 6).
edge(d, e, 5).
```

(a) (3 Pontos) Escreva um predicado `fourPath([A, B, C, D])` que seja verdadeiro se houver um caminho no grafo formado pelos nós `A`, `B`, `C` e `D`. Por exemplo, para o grafo acima, obteríamos:

```
?- fourPath(L).
L = [a, b, c, a] ;
L = [a, b, c, d] ;
L = [a, c, a, b] ;
L = [a, c, a, c] ;
L = [a, c, d, e] ;
L = [b, c, a, b] ;
L = [b, c, a, c] ;
L = [b, c, d, e] ;
L = [c, a, b, c] ;
L = [c, a, c, a] ;
L = [c, a, c, d] ;
```

(b) (3 Pontos) Escreva um predicado `countFourPath(N)`, que seja verdade quando `N` for o número de caminhos de quatro nós no grafo formado pelos predicados `edge`. Você pode assumir que seu predicado `fourPath` da questão anterior está corretamente implementado. Exemplo:

```
?- countFourPath(N).
N = 11.
```

(c) (4 Pontos) Escreva um predicado `notEuc(X, Y, Z)`, que seja verdade se houver um triângulo não euclidiano no grafo. Um triângulo é euclidiano se a soma de quais dois lados é maior que o tamanho do terceiro lado. Assuma que o terceiro elemento de cada átomo `edge(A, B, T)` é o tamanho da aresta formada pelos nós `A` e `B`. Por exemplo:

```
?- notEuc(X, Y, Z).
X = a,
Y = b,
Z = c
```

5. Cada um dos itens abaixo contém uma construção Java. Em cada uma dessas situações, diga se uma exceção pode acontecer, e, em caso afirmativo, explique que tipo de exceção é essa. Não é necessário dizer o nome da exceção. Explique somente o tipo de erro que ela descreve. Por exemplo, não é preciso escrever `SonBeatMotherException`; em vez disso, você pode escrever: "esse erro ocorre quando um filho usa de violência contra a sua própria mãe".

   (a) (2 Pontos) Invocação de método em Java:

   ```
   Object o = getNewObject();
   o.toString();
   ```

   (b) (2 Pontos) Multiplicação de inteiros em Java:

   ```
   int a = 1073741824; // 2^30
   int b = 1073741825; // 2^30 + 1
   int c = a * b;
   ```

   (c) (2 Pontos) Coerção em Java:

   ```
   Dog d = (Dog)x;
   ```

   (d) (2 Pontos) Acesso ao primeiro elemento de um arranjo em Java:

   ```
   int x = v[0];
   ```

   (e) (2 Pontos) Adição de `double` e `byte` em Java:

   ```
   double d = 3.141596;
   byte b = 3;
   double c = b + d;
   ```