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:

- 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) Podemos representar estruturas de dados em SML via tipos algébricos. A declaração abaixo, por exemplo, descreve uma árvore binária:

```
datatype 'a tree = Empty | Node of 'a tree * 'a * 'a tree;
```

Implemente uma função `mr`, de tipo `('a -> 'b) -> 'a tree -> 'b tree` que mapeie uma árvore em outra. As chamadas abaixo devem ser válidas para a sua implementação:

```
- val t0 = Node ((Node (Empty, 4, Empty)), 3, (Node (Empty, 5, Empty)));
val t0 = Node (Node (Empty,4,Empty),3,Node (Empty,5,Empty)) : int tree
- mr (fn x => x * 2) t0;
val it = Node (Node (Empty,8,Empty),6,Node (Empty,10,Empty)) : int tree
- val t1 = Node ((Node (Empty, "a", Empty)), "b", (Node (Empty, "c", Empty)));
val t1 = Node (Node (Empty,"a",Empty),"b",Node (Empty,"c",Empty))
  : string tree
- mr (fn x => ".." ^ x ^ "..") t1;
val it = Node (Node (Empty,"..a..",Empty),"..b..",Node (Empty,"..c..",Empty))
  : string tree
```

2. Para cada uma das chamadas de função abaixo, diga se a chamada é válida, ou se a chamada não é possível. No caso da chamada ser válida, escreva seu resultado. No caso da chamada não ser possível, explique o porquê.

   (a) (2 pontos)

   ```
   datatype exint = Value of int | PlusInf | MinusInf;
   map Value [1, 2, 3, 4]
   ```

   (b) (3 pontos)

   ```
   foldl (fn(x, y) => x ^ y) "" ["a", "b", "c"]
   ```

   (c) (2 pontos)

   ```
   fun f g h = g (g h)
   f (fn x => x * x) 3
   ```

   (d) (3 pontos)

   ```
   fun f g h = g g h
   f (fn x => x * x) 3
   ```

3. Considere a linguagem da emoção, que possui dois tipos de símbolos: nomes e verbos. Nomes são elementos do conjunto {joao, maria, jose, ana, antonio, rita}, e verbos são elementos do conjunto {ama, odeia}. Frases na linguagen da emoção podem ser do tipo:

$$nome\ verbo\ nome\ (que\ verbo\ nome)*$$

Sendo o símbolo * usado para indicar zero ou mais repetições do padrão. Escreva uma gramática lógica em Prolog que reconheça a linguagem do amor. Alguns exemplos de buscas, usando essa gramática lógica, são dados logo abaixo:

```
?- quem([joao], []).
false.

?- quem([joao, ama, maria], []).
true ;
false.

?- quem([joao, ama, maria, que], []).
false.

?- quem([joao, ama, maria, que, ama, antonio], []).
true ;
false.

?- quem([joao, ama, maria, que, odeia, maria, que, ama, jose], []).
true ;
false.

?- quem([joao, ama, joao, que, ama, joao, que, odeia, joao], []).
true ;
false.
```

4. O objetivo dessa questão é desenvolver alguns programas simples em SML para a manipulação de listas.

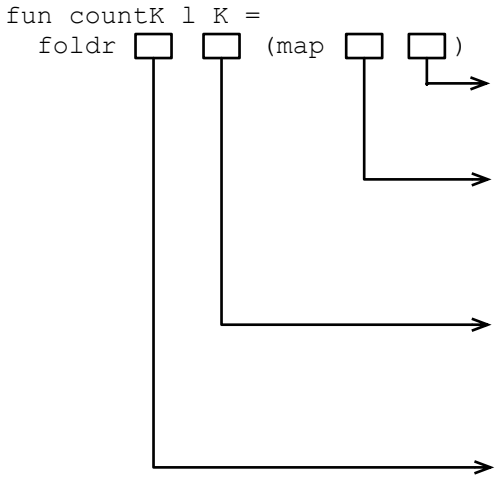
   (a) (5 pontos) Implemente a função `length` de tipo `'a list -> int`, que informe o tamanho de uma lista. Note que essa função é parte da biblioteca padrão de SML. Você não deve usar a função que já existe; implemente uma versão recursiva a partir de operadores mais básicos de manipulação de listas. Por exemplo:

```
- length nil;
val it = 0 : int
- length [1,2,3];
val it = 3 : int
- length [ [1,2,3], [4,5,6], [7,8,9] ];
val it = 3 : int
```

   (b) (5 pontos) Use a implementação de `length` criada no exercício anterior para implementar uma função `countK`, de tipo `'a list list -> int -> int`, que conte o número de listas de tamanho K presentes em uma lista de listas. Por exemplo:

```
- countK [[1], [1,2], [2], [3, 4], [5]] 1;
val it = 3 : int
- countK [ ["a", "b"], ["a", "b", "c"], ["a"] ] 3;
val it = 1 : int
```

   A sua implementação deve fazer uma chamada de `foldr` e outra de `map`, de acordo com o modelo abaixo:



   Uma última dica:

```
- map;
val it = fn : ('a -> 'b) -> 'a list -> 'b list
- foldr;
val it = fn : ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
```

5. Essa questão refere-se ao programa abaixo. Por incrível que pareça, quando compilado com `gcc 4.2` no sistema operacional Mac OS X, Versão 10.5.8, esse programa não termina.

```
void foo(int param) {
  int v[1];
  int i = 0;
  for (; i < param; i++) {
    v[i] = v[i] - 4;
  }
}

int main() { foo(2); }
```

   (a) (5 pontos) Que suposições você poderia fazer acerca do registro de ativação desse programa para explicar a não terminação?

   (b) (3 pontos) Esse *loop* infinito não aconteceria caso C possuísse o mesmo sistema de tipagem forte de Java. Por que?

   (c) (2 pontos) Cite uma vantagem do sistema de tipagem fraca de C sobre o sistema de tipagem forte de Java.

6. Essa questão refere-se aos dois programas abaixo, e suas respectivas saídas. Um dos programas foi escrito em *bash script*, e o outro foi escrito em C:

scope.bash

```
#!/bin/bash
x=1

function g () {
  echo "Dentro de g(), x = $x" ;
  x=2 ;
}

function f () {
  local x=3 ;
  g ;
}

f
echo "Ultimo valor de x = $x"
```

saída de scope.bash

```
Dentro de g(), x = 3
Ultimo valor de x = 1
```

scope.c

```
int x = 1;

void g() {
  printf("Dentro de g(), x = %d\n", x);
  x = 2;
}

void f() {
  int x = 3;
  g();
}

int main() {
  f();
  printf("Ultimo valor de x = %d\n", x);
}
```

saída de scope.c

```
Dentro de g(), x = 1
Ultimo valor de x = 2
```

(a) (1 pontos) Essas duas linguagens, *bash script* e C, possuem formas de escopo diferentes. Uma delas possui escopo dinâmico, e a outra escopo estático. Qual delas possui escopo estático?

(b) (5 pontos) Qual a diferença entre esses dois mecanismos de escopo?

(c) (4 pontos) Linguagens mais modernas, como Java e C# tendem a ser projetadas com escopo estático. Porque o escopo dinâmico está caindo em desuso? Explique qual a desvantagem desse tipo de escopo.