# Primeira Prova de Linguagens de Programação
# - DCC024 -
# 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.

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.
- Lembre-se do P.O.F. da aula quatro. E lembre-se também: perguntando qual o P.O.F. da aula quatro, você perde a sua pergunta.
- A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados

| Questão 1 | Questão 2 | Questão 3 | Questão 4 | Questão 5 | Questão 6 |
|---|---|---|---|---|---|
| | | | | | |

1. Esta questão diz respeito à gramática abaixo, que representa expressões booleanas em SML:

$$\begin{array}{rcl} E & ::= & E \text{ andalso } E \\ & | & E \text{ orelse } E \\ & | & \text{not } E \end{array}$$

(a) (3 pontos) Esta gramática é ambígua ou não? Justifique a sua resposta.

(b) (5 pontos) Modifique a gramática para que o operador not tenha a maior precedência, e o operador orelse tenha a menor.

(c) (2 pontos) Qual é a associatividade do operador not na gramática original?

2. Considere as duas funções abaixo, que calculam o fatorial de um número inteiro. O primeiro programa, escrito em SML, é dado logo abaixo:

```
fun fact n =
  if n > 1
  then n * fact (n-1)
  else 1
```

O segundo programa, escrito em JavaScript, não é muito diferente:

```
fact(n) {
  if (n > 1)
    return n * fact(n - 1);
  else
    return 1;
}
```

Nenhum destes programas usa declarações de tipos. Por outro lado, sendo SML uma linguagem estaticamente tipada, e JavaScript uma linguagem dinamicamente tipada, estas linguagens usam sistemas de tipos muito diferentes.

(a) (2 pontos) Como os tipos são descobertos em SML?

(b) (2 pontos) Como os tipos são descobertos em JavaScript?

(c) (3 pontos) Cite uma vantagem da tipagem estática sobre a tipagem dinâmica.

(d) (3 pontos) Cite uma vantagem da tipagem dinâmica sobre a tipagem estática.

3. Podemos usar uma tupla como um registro em SML. Por exemplo, para representar uma entrada em um banco de dados, podemos usar uma tupla de três elementos `(name, age, salary)`, de tipo `(string * int * real)`, em que o primeiro campo representa um nome, o segundo uma idade, e o terceiro um salário. Uma relação de três pessoas poderia ser representada pela lista abaixo:

   ```
   val minhaLista = [("f",31,570.0),("p",21,200.0),("g",27,900.0)]
   ```

   Todas as implementações a seguir devem ser feitas em SML:

   (a) (3 pontos) Escreva uma função `sum`, de tipo `('a * 'b * real) list -> real`, que some todos os salários na lista de tuplas. Por exemplo:

   ```
   - sum minhaLista
   val it = 1670.0
   ```

   Lembre-se, o salário é representado por números **reais**. O operador de soma, em SML, por padrão, espera números inteiros. Anotações de tipo devem ser usadas para modificar este comportamento.

   (b) (4 pontos) Escreva uma função `oldest`, de tipo `('a * int * 'b) list -> int`, que encontre a idade do elemento de maior idade em uma lista de tuplas. Por exemplo:

   ```
   - oldest minhaLista
   val it = 31
   ```

   (c) (3 pontos) Escreva uma função `filter30`, de tipo `('a * int * 'b) list -> 'a list`, que retorne uma sublista da lista de entrada contendo o nome de todos os elementos da lista original cuja idade seja maior que ou igual a 30. Por exemplo:

   ```
   - filter30 minhaLista
   val it = ["f"]
   ```

4. Esta questão refere-se aos dois programas abaixo, escritos em SML/NJ e C:

```
val one = 1;

fun inc x = x + one;

fun addTwo y =
  let
    val one = 2
  in
    inc y
  end;

addTwo 4;
```

```
#include <stdio.h>

int one = 1;

int inc(int x) { return x + one;}

int addTwo(int y) {
  one = 2;
  return inc(y);
}

int main() {
  printf("%d\n", addTwo(4));
}
```

(a) (2 pontos) Qual será o valor retornado pela chamada `addTwo 4` no programa escrito em SML/NJ?

(b) (2 pontos) Qual o valor impresso pelo programa escrito em C?

(c) (3 pontos) Com base em sua resposta na letra (a), SML/NJ usa escopo estático ou dinâmico?

(d) (3 pontos) Qual valor seria retornado por `addTwo 4` caso SML/NJ usasse o outro tipo de escopo? Existem dois tipos de escopo: estático ou dinâmico. Você deve ter respondido um deles na questão anterior. Logo, suponha o outro nesta questão.

5. (2 pontos cada) Escreva o tipo de cada uma das declarações do nome `f` abaixo:

   (a) `fun f(x) = x`

   (b) `fun f(g, x) = g(x)`

   (c) `val f = map (fn x => x + 1)`

   (d) `val f = foldr`

   (e) `val f = foldl (fn(a, b) => a ^ b)`

6. Esta questão refere-se à função abaixo, implementada em SML:

```
fun foo n =
  let
    fun bar 0 = nil
      | bar m = n :: bar (m - 1)
  in
    bar n
  end
```

(a) (1 ponto) Qual o valor retornado pela chamada `foo 2`?

(b) (3 pontos) SML utiliza um espaço em memória chamado *registro de ativação* para guardar as informações necessárias à execução de uma função. Cite pelo menos quatro informações diferentes que precisam ser guardadas no registro de ativação de uma função em SML.

(c) (3 pontos) Ao contrário de SML, os registros de ativação de ANSI C não armazenam um *link de aninhamento*. Por que?

(d) (3 pontos) Desenho os registros de ativação criados durante a chamada `foo 2`.