Primeira Prova de Linguagens de Programação
- DCC024W -
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.
- 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) A gramática abaixo reconhece expressões com parênteses balanceados:

$$\begin{array}{rcl} \langle E \rangle & ::= & '(' \ \langle E \rangle \ ')' \ \langle E \rangle \\ & | & \epsilon \end{array}$$

O símbolo ϵ representa a cadeia vazia. Implemente uma gramática lógica em Prolog que reconheça a mesma linguagem, e que conte o número de parênteses abertos. Por exemplo, assumindo-se que a gramática foi implementa em um arquivo chamado `tree_depth`, as seguintes buscas devem ser válidas:

```
?- consult(tree_depth).
% tree_depth compiled 0.01 sec, 260 bytes
true.

?- tree(D, ['(', ')'], []).
D = 1 ;
false.

?- tree(D, ['(', '(', ')', ')', '(', ')'], []).
D = 3 ;
false.

?- tree(D, ['(', ')', '(', '(', ')', ')', '(', ')'], []).
D = 4 ;
false.
```

2. (10 Pontos) Implemente, em SML, a função `stripCommas`, cujo tipo é `string -> string`. Essa função recebe uma *string* e retorna outra *string* que é uma cópia da primeira, exceto que todas as vírgulas foram removidas. Por exemplo:

```
- stripCommas "o pato, que eh legal, fugiu!";
val it = "o pato que eh legal fugiu!" : string
-
- stripCommas ",,,,,";
val it = "" : string
```

Você pode achar útil, para esse exercício, usar as funções predefinidas `implode` e `explode`. A primeira, de tipo `char list -> string`, recebe uma lista de caracteres, e produz uma *string* com esses caracteres. A segunda função, de tipo `string -> char list`, converte uma *string* para uma lista de caracteres.

3. (10 Pontos) A única operação que podemos realizar sobre tuplas, em SML, é a *indexação*. Essa é a operação que nos permite ler o campo de uma tupla:

```
- val pair = (13, 29);
val pair = (13,29) : int * int
- #1 pair;
val it = 13 : int
- #1 pair + #2 pair;
val it = 42 : int
```

SML permite que somente literais inteiros sejam usados como índices de tuplas. As seguintes chamadas resultam em erros:

```
- (* Uso de uma expressao aritmetica como indice de tupla *)
- #(1 + 1) pair;
stdIn:5.2-5.4 Error: syntax error: deleting  LPAREN INT
-
- (* Uso de uma variavel como indice de tupla *)
- val x = 1;
val x = 1 : int
- #x pair;
stdIn:6.1-6.8 Error: operator and operand don't agree [record labels]
```

Qual a razão por trás dessa restrição de SML? Em outras palavras, quando projetando a sua linguagem, por que *Robin Milner* decidiu não permitir que expressões gerais de tipo inteiro pudessem ser usadas para indexar tuplas?

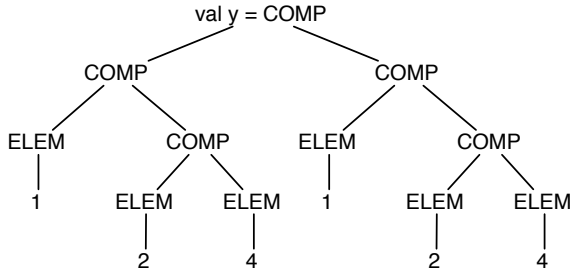
4. (10 pontos) A linguagem SML não permite que o usuário defina símbolos sobrecarregados, ao contrário de linguagens tais como C++ e Java. Por outro lado, a seguinte sequência de comandos é perfeitamente válida em SML:

```
- fun op + (a, b) = a - b;
val + = fn : int * int -> int
- 3 + 4;
val it = ~1 : int
```

Por que não podemos dizer que o símbolo '+' foi sobrecarregado nesse caso?

5. Essa questão refere-se ao tipo algébrico `composite`, que pode representar um elemento polimórfico, ou uma tupla de dois `composite`s. Para um exemplo, veja as declarações abaixo, que criam duas variáveis do tipo `composite`, x e y:

```
val x = COMP (ELEM 1, COMP (ELEM 2, ELEM 4))
val y = COMP (x, x)
```



(a) (3 pontos) Complete a declaração de `composite` abaixo:

```
- datatype 'a composite =
```

(b) (5 pontos) Crie uma função `sum` do tipo `int composite -> int` que retorne a soma de todos os elementos em uma estrutura do tipo `int composite`.

(c) (1 ponto) Que forma de polimorfismo é utilizada pelo tipo `composite`?

(d) (1 ponto) A linguagem C possui a forma de polimorfismo que você respondeu na questão anterior?

6. Esta questão refere-se ao programa abaixo, escrito na linguagem C:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  typedef struct ThreeDPointType { int x; int y; int z; } ThreeDPoint;
4  typedef struct DateType { int day; int month; int year; } Date;
5  int main() {
6    ThreeDPoint *p = (ThreeDPoint*)malloc(sizeof(ThreeDPoint));
7    Date *d;
8    char a[] = {7, 0, 0, 0, 7, 0, 0, 0, 30, 7, 0, 0};
9    p->x = 7;
10   p->y = 7;
11   p->z = 1822;
12   d = (struct ThreeDPoint*)p;
13   printf("%d, %d, %d\n", d->day, d->month, d->year);
14   d = (struct ThreeDPoint*)a;
15   printf("%d, %d, %d\n", d->day, d->month, d->year);
16 }
```

Esse programa, se compilado e executado, imprimirá a seguinte saída:

```
7, 7, 1822
7, 7, 1822
```

(a) (5 pontos) Esse programa evidencia que C é uma linguagem fracamente tipada. Por que?

(b) (2 pontos) O resultado impresso depende do compilador utilizado para compilar o programa? Justifique a sua resposta.

(c) (3 pontos) Esse tipo de contrução sem dúvida torna os programas escritos em C mais difíceis de serem entendidos. Ainda assim essas coerções inseguras são úteis. Descreva uma situação em que você poderia usar esse tipo de padrão de programação.