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. Todas as respostas abaixo devem ser justificadas com base nas observações a seguir. Na linguagem de programação C, a expressão `2 == 1 + 1` é sintaticamente válida, e seu valor é 1. Em SML a expressão equivalente `2 = 1 + 1` também é válida. Em C, a expressão `(2 == 1) + 1` também é válida, mas em SML a expressão equivalente `(2 = 1) + 1` não é sintaticamente aceitável.

   (a) (2 Pontos) Na linguagem de programação C, dentre os operadores ==, e +, qual deles possui maior precedência?

   (b) (2 Pontos) Por que a expressão `(2 = 1) + 1` não é válida em SML?

   (c) (2 Pontos) Qual é a precedência relativa entre os operadores = e + em SML?

   (d) (2 Pontos) A expressão `(2 = 1) + 1` é válida em C? Em caso afirmativo, qual o seu valor? Em caso negativo, justifique.

   (e) (2 Pontos) Por que as expressões `2 == 1 + 1` e `(2 == 1) + 1` não geram os mesmos valores em C?

2. (10 Pontos) O problema de mostrar que uma gramática é ambígua, ou que ela não é ambígua, é indecidível em geral. Por outro lado, é possível mostrar que algumas gramáticas simples não são ambíguas. Considere a gramática abaixo:

i	$\langle S \rangle$	$::=$	$\langle G \rangle \ \langle N \rangle$
ii		$\mid$	$\langle N \rangle$
iii	$\langle G \rangle$	$::=$	$+ \mid -$
iv	$\langle N \rangle$	$::=$	$\langle D \rangle \ \langle N \rangle$
v		$\mid$	$\langle D \rangle$
vi	$\langle D \rangle$	$::=$	1

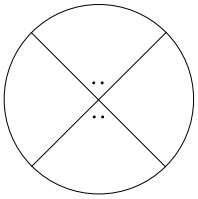
Prove que essa gramática não é ambígua. Para lhe auxiliar na explicação da prova, as regras de produção foram numeradas.

3. (10 Pontos) Considere o programa abaixo, escrito em Python:

```
>>> def f(x):
...   if x:
...     return f
...
>>> f(True)
<function f at 0x652b0>
```

Esse programa implementa uma função `f` que retorna a si mesma. É possível escrever uma função, em SML, que retorne a si mesma? Se sua resposta for afirmativa, demonstre-a com um exemplo. Se sua resposta for falsa, explique porque não é possível escrever tal função.

1 Ponto Extra: o que significa o desenho abaixo?

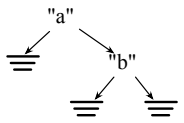


4. (10 Pontos) Considere a definição de uma árvore binária em SML:

```
datatype 'data tree =
   Empty |
   Node of 'data tree * 'data * 'data tree
```

O restante desta questão refere-se a essa definição:

(a) (2 Pontos) Declare a árvore abaixo, em SML, isto é, `val t = ...`:



(b) (3 Pontos) A função abaixo converte uma árvore para uma lista:

```
fun listall Empty = nil
  | listall (Node (x, y, z)) = listall x @ y :: listall z;
```

Qual é o tipo de função `listall`?

(c) (3 Pontos) A função abaixo é tal que `foldTree f x t = foldr f x (listall t)`. Qual é o tipo de `foldTree`?

```
fun foldTree f x Empty = x
  | foldTree f x (Node (t1, y, t2)) =
    let
      val v1 = foldTree f x t2
      val v2 = f(y, v1)
      val v3 = foldTree f v2 t1
    in
      v3
    end
```

(d) (2 Pontos) A implementação de `dup`, abaixo, compara `foldTree` e `listall`, retornando uma dupla com os resultados que essas duas funções produzem:

```
fun dup f s t = (foldr f s (listall t), foldTree f s t)
```

Qual é o tipo de `dup`?

5. Esta questão refere-se a seguinte expressão lâmbda: $(\lambda x.\lambda y.yx)\ y\ (\lambda x.x)\ y$

   (a) (5 Pontos) Encontre a forma normal dessa expressão, mostrando, explicitamente, todos os passos de sua resolução.

   (b) (1 Ponto) A associatividade de aplicação de funções no cálculo lâmbda é a mesma que a associatividade de aplicações de funções em SML. Essa associatividade é da esquerda-para-a-direita, ou da direta-para-a-esquerda? No primeiro caso, temos que $e_1\ e_2\ e_3 = (e_1\ e_2)\ e_3$, no segundo, temos que $e_1\ e_2\ e_3 = e_1\ (e_2\ e_3)$.

   (c) (4 Pontos) Se a associatividade de aplicações de funções no cálculo lâmbda fosse o contrário do que você respondeu na questão anterior, qual seria a forma normal de $(\lambda x.\lambda y.yx)\ y\ (\lambda x.x)\ y$? Novamente, faça todas as substituições explicitamente, assumindo a associatividade diferente.

6. O registro de ativação de uma função é a área de memória que armazena todas as informações necessárias para o correto funcionamento daquela função. As três questões abaixo referem-se à esses registros de ativação. Em cada questão, você deve fornecer um pequeno exemplo de código em SML, e explicar porque o seu exemplo atende o que se pede na questão.

   (a) (3 Pontos) Escreva o mais curto exemplo, em SML/NJ em que você consiga pensar, que não funcionaria se implementado usando um registro de ativação alocado estaticamente. Explique porque este exemplo falharia.

   (b) (3 Pontos) Escreva o mais curto exemplo, em SML/NJ, em que você consiga pensar, que não funcionaria corretamente caso SML/NJ não implementasse registros de ativação usando links de aninhamento. Isto é, os registros de ativação podem ser implementados via uma pilha, como é normalmente feito, mas não possuem link de aninhamento.

   (c) (4 Pontos) Escreva o mais curto exemplo que você consiga pensar que não funcionaria se todos os dados armazenados no registro de ativação de uma função f, em SML, fossem desalocados imediatamente após o retorno de f.