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. Esta questão refere-se às duas funções abaixo, implementadas em SML:

```
fun primeiroLista [a, _] = a

fun primeiroTupla (a, b) = a;
```

   (a) (2 Pontos) Qual é o tipo inferido para a função `primeiroLista`?

   (b) (3 Pontos) Qual é o tipo inferido para a função `primeiroTupla`?

   (c) (5 Pontos) Ao compilarmos a função `primeiroLista` obtemos um aviso de casamento de padrões não exaustivo (`Warning: match nonexhaustive`). Por outro lado, esse aviso não aparece ao compilarmos `primeiroTupla`. Por que o aviso acontece no caso de `primeiroLista`, e não acontece no caso de `primeiroTupla`?

2. Considere a gramática lógica abaixo, que reconhece cadeias de zero ou mais ocorrências dos símbolos `+` e `-`:

```
mp --> [+], mp.
mp --> [-], mp.
mp --> [].
```

   (a) (2 Pontos) Essa gramática não é ambígua. Escreva, usando a notação BNF, uma gramática que reconheça a mesma linguagem, mas que seja ambígua.

   (b) (2 Pontos) Demonstre que a gramática que você criou na questão anterior é, de fato ambígua.

   (c) (6 Pontos) Modifique a gramática lógica mostrada no início desta questão, adicionando-lhe um atributo `D`. Esse atributo deve informar a diferença entre símbolos `+` e símbolos `-`. Por exemplo:

```
?- mp(D, [], []).
D = 0.

?- mp(D, [-, +, +], []).
D = 1 ;
false.

?- mp(D, [-, -, +], []).
D = -1 ;
false.
```

3. Em cada um dos programas abaixo, temos a execução de um comando de atribuição. Explique o que acontecerá em cada caso. As respostas possíveis são: (i) o comando de atribuição impede a compilação do programa; (ii) o programa compila, mas um erro ocorre em tempo de execução e (iii) o programa compila, e um erro não - necessariamente - acontece em tempo de execução; (iv) o programa está correto. Você deve justificar sua resposta citando alguns dentre esses pontos: (a) tipagem forte vs tipagem fraca; (b) equivalência estrutural vs equivalência nominal.

(a) 2 Pontos - C

```
struct point { int x, y; };
struct triple { int p1, p2, p3; };
int main() {
  struct point* p = (struct point*)malloc(sizeof(struct point));
  struct triple* t = (struct triple*)malloc(sizeof(struct triple));
  t->p1 = t->p2 = t->p3 = 1;
  p = (struct point*)t;
  printf("%d, %d\n", p->x, p->y);
}
```

(b) 2 Pontos - Java

O uso de extends faz com que o sistema de tipos não barre essa atribuição

```
class Point { int x, y; }
class Triple extends Point { int p1, p2, p3; }
public static void main(String args[]) {
   Point p = new Point();
   Triple t = new Triple();
   t.p1 = t.p2 = t.p3 = 1;
   p = (Triple)t;
   System.out.println("Resultado = " + p.x);
}
```

(c) 3 Pontos

```
type Point = {x:int, y:int};
type Pair = {x:int, y:int};
val p:Point = {x = 4, y = 5};
val p = {x=4,y=5} : Point
val t = {x=4,y=5} : Pair
```

(d) 3 Pontos

```
struct point { int x, y; };
struct pair { int x, y; };
int main() {
  struct point p;
  struct pair t;
  p = t;
}
```

4. Seja `mapFun` uma função, em SML, que recebe um valor v e uma lista de funções l. Dadas essas entradas, `mapFun` retorna uma nova lista que resulta da aplicação de cada função em l a v. Exemplos são mostrados logo abaixo:

```
- mapFun 2 [fn x => x + 1, fn x => x * x];
val it = [3,4] : int list
- mapFun true [fn x => x orelse false, fn y => not y];
val it = [true,false] : bool list
- mapFun 3.14 [fn x => x * x, fn y => y + 2.76, fn z => ~z, fn w => w / 3.14];
val it = [9.8596,5.9,~3.14,1.0] : real list
```

   (a) (2 Pontos) Qual é o tipo que o compilador de SML vai inferir para `mapFun`?

   (b) (6 Pontos) Implemente a função `mapFun` em SML.

   (c) (2 Pontos) Sendo v e l o valor e a lista de entrada, qual é a complexidade asimptótica de sua implementação de `mapFun`, criada na questão anterior, assumindo que cada função em l é $O(1)$?

5. A linguagem SML possui escopo estático. Na verdade, muito poucas linguagens de programação ainda em uso possuem o escopo dinâmico. A impopularidade desse tipo de escopo deve-se a uma série de desvantagens que ele possui, e nessa questão analisaremos algumas delas.

   (a) (1 Ponto) Lembrando que SML possui escopo dinâmico, qual é o resultado de `foo 5` no programa abaixo?

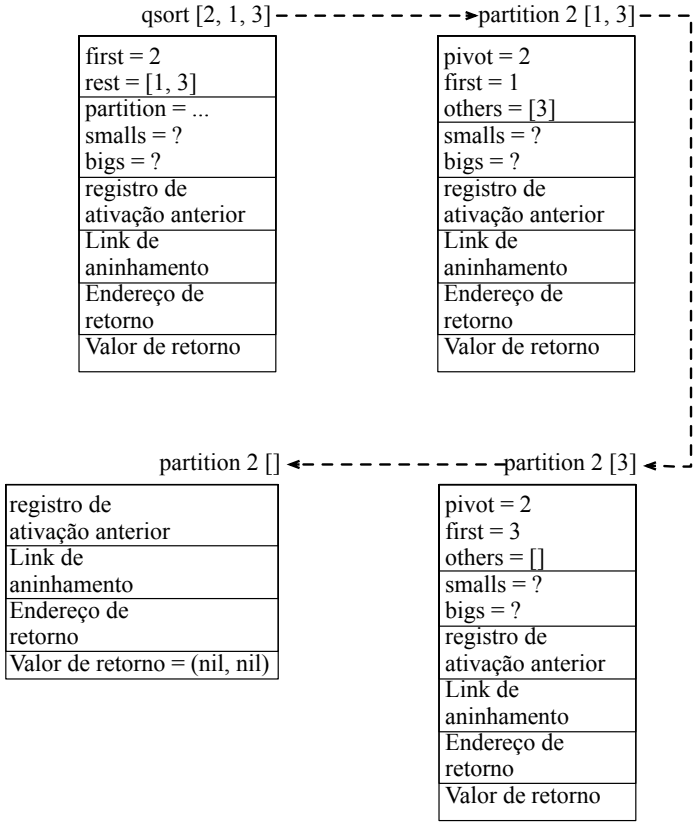
   ```
   val process = fn x => x + 1;

   fun inc x = process x;

   fun foo x = let val process = fn x => x - 1; in inc x end;

   foo 5;
   ```

   (b) (3 Pontos) Se SML possuísse escopo dinâmico, então qual seria o resultado de `foo 5` no programa acima? Justifique a sua resposta explicando como o escopo dinâmico funciona para esse exemplo em particular.

   (c) (6 Pontos) Cite e explique uma das desvantagens do escopo dinâmico em relação ao escopo estático. Sinta-se livre para usar o programa mostrado nesta questão como um exemplo.

6. Essa questão refere-se ao algoritmo `qsort`, cuja implementação, mais os registros de ativação para a chamada `qsort [2, 1, 3]` são mostrados logo abaixo. Note que as setas tracejadas servem apenas para mostrar a ordem em que ocorreram as chamadas:

```
fun qsort nil = nil
 | qsort (first::rest) =
    let
     fun partition (pivot, nil) = (nil,nil)
      | partition (pivot, first :: others) =
          let
           val (smalls, bigs) =
            partition(pivot, others)
          in
           if first < pivot
            then (first::smalls, bigs)
            else (smalls, first::bigs)
          end;
     val (smalls, bigs) = partition(first,rest)
    in
     qsort(smalls) @ [first] @ qsort(bigs)
    end;
```



(a) (2 Pontos) Desenhe linhas tracejadas entre os endereços de retorno e os pontos para onde eles apontam.

(b) (2 Pontos) Desenhe linhas contíguas entre os ponteiros "*registro de ativação anterior*", e a memória para onde eles apontam.

(c) (2 Pontos) Desenhe linhas pontilhadas entre os ponteiros "*link de aninhamento*"e a área de memória para onde eles apontam. Tome cuidado para que suas linhas pontilhadas não se confundam com as linhas tracejadas do item (6.a).

(d) (4 Pontos) Qual funcionalidade de SML (que não existe, por exemplo, em ANSI C [1]) faz com que sejam necessários links de aninhamento em registros de ativação?

[1] Mas que o gcc implementa...