Primeira Prova de Linguagens de Programação
- DCC024B -
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:

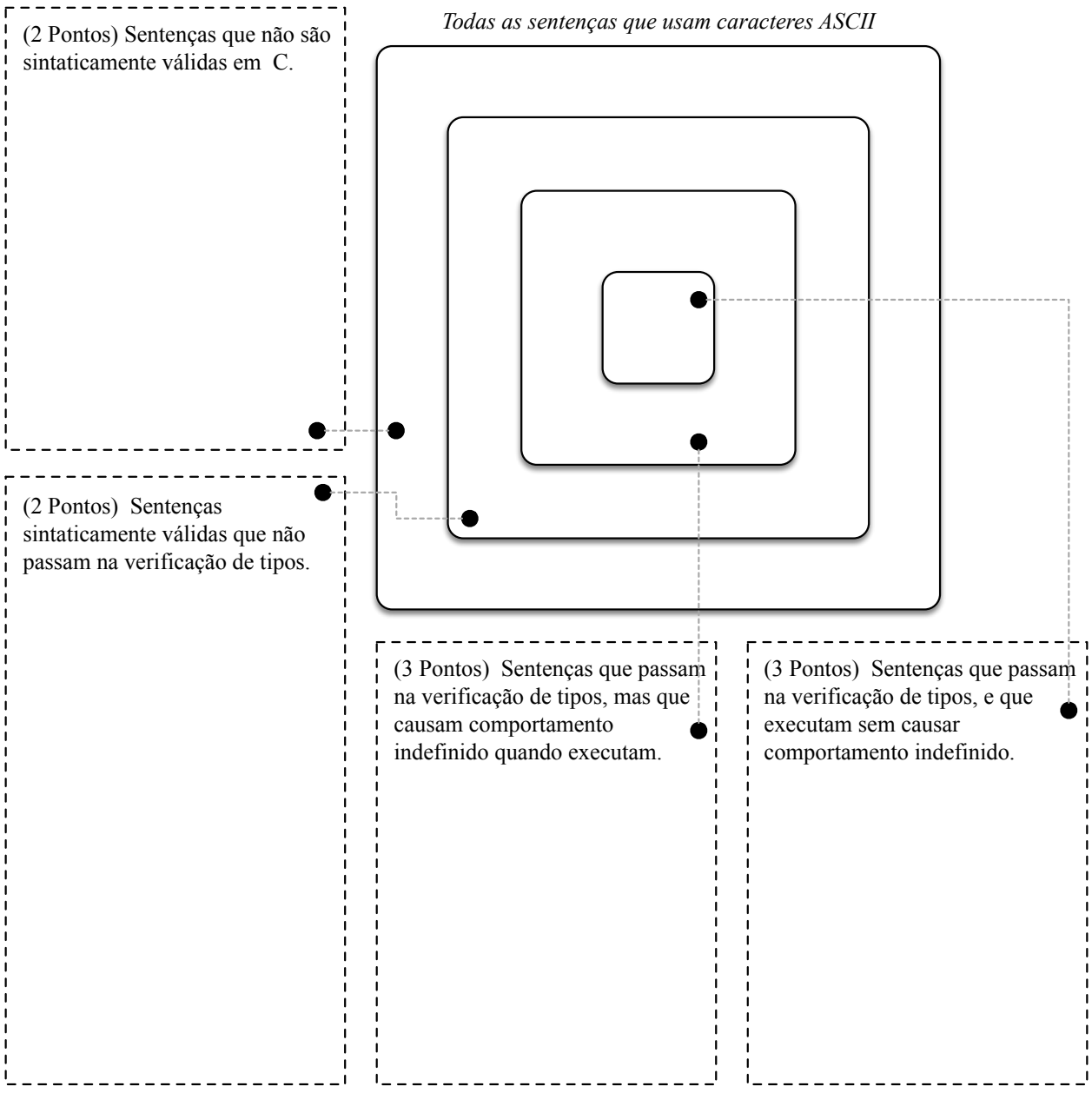
1. Escreva sempre algo nas questões, a fim de ganhar algum crédito parcial.
2. Se não entender a questão, e já tiver gasto sua pergunta, escreva a sua interpretação da questão junto à resposta.
3. A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Extra

Questão extra (0.5): "*O meu nome é Dejair, facinho de confundir com João do Caminhão*". Esta é uma das frases mais geniais jamais escritas em música popular brasileira. Qual o nome da música em que essa frase originalmente apareceu?

1. Um programa é uma sequência de caracteres. Se tal sequência é reconhecida pela gramática de uma linguagem de programação, então o programa é sintaticamente válido naquela linguagem. Porém, nem todo programa sintaticamente válido é semanticamente válido. O diagrama abaixo ilustra essas relações de validade para a linguagem de programação C. Complete as caixas com exemplos de programas que pertencem à cada categoria marcada no Diagrama de Venn.



2. Nesta questão você deverá escrever diferentes programas em SML/NJ. Lembre-se que a correta sintaxe é importante. Para cada programa, você precisa usar o número de linhas e colunas especificado.

   (a) (3 Pontos) Escreva uma função `inv`, cujo tipo é `'a list -> 'a list`, que inverta pares de caracteres dentro de uma lista. Exemplos:

```
inv [1, 2, 3, 4];
val it = [2,1,4,3] : int list

inv ["a", "b", "c"];
val it = ["b","a","c"] : string list

inv nil:int list;
val it = [] : int list
```

   Escreva seu programa aqui, um caractere por quadro:

   Veja que `inv` inverte os caracteres em pares, isso é, nas posições $2n$ e $2n+1, n \geq 0$. A função `inv` não inverte os caracteres nas posições $2n+1$ e $2n+2, n \geq 0$.

   (b) (3 Pontos) Escreva uma função `xch`, que troque de lugar o primeiro e o terceiro elementos de uma tupla de três elementos. Exemplos:

```
- xch ("oi", true, 1);
val it = (1,true,"oi") : int * bool * string
- xch (3.14, #"a", false);
val it = (false,#"a",3.14) : bool * char * real
```

   Escreva seu programa aqui, um caractere por quadro:

   (c) (2 Pontos) Qual é o tipo da função `xch`?

   (d) (2 Pontos) qual é o resultado da seguinte chamada?

```
xch (3.14, #"a", false, 314);
```

3. A questão abaixo refere-se ao seguinte programa, implementado em C. Você precisa entender o que o programa faz. O desafio da questão consiste em simular esse programa em SML/NJ. O programa análogo, em SML/NJ, não vai ser exatamente igual ao programa em C. Por exemplo, em vez de usar um arranjo, usaremos uma lista, e o tamanho do arranjo não será um parâmetro da nova função.

```
int incs(int* v, int N) {
  int sum = 0;
  for (int i = 1; i < N; i++) { if (v[i] > v[i-1]) { sum++; } }
  return sum;
}
```

(a) (4 Pontos) Implemente uma função `zip`, que receba duas listas, `L0` e `L1`, e retorne uma lista de pares formados com os elementos em posições correspondentes de `L0` e `L1`. O tipo de `zip` deve ser `'a list -> 'b list -> ('a * 'b) list`. Exemplos:

```
- zip ["a", "b"] [1, 2, 3];
val it = [("a",1),("b",2)] : (string * int) list
- zip [true, false, true] (nil:int list);
val it = [] : (bool * int) list
```

Escreva seu programa aqui, um caractere por quadro:

(b) (3 Pontos) Implemente uma função `toIncs`, de tipo `int list -> int list`, que mapeia uma lista de inteiros `L` para uma lista `B`, de booleanos. O elemento n de `B` é `true` se o elemento na posição n de `L` for maior que o elemento na posição $n-1, n > 0$ de `L`. Você pode reusar a função `zip`. Se não a tiver feito, você pode assumir sua existência. Você também pode reusar a função `tl`, vista em sala de aula. Exemplos:

```
- toIncs [3, 2, 3, 3, 1, 4];
val it = [false,true,false,false,true] : bool list (* 3 < 2, 2 < 3, 3 < 3, ..., 1 < 4 *)
- toIncs [];
exception match non exhaustive (* nao se preocupe com o erro. *)
- toIncs [1];
val it = [] : bool list
```

Escreva seu programa aqui, um caractere por quadro:

(c) (3 Pontos) Escreva uma função `incs`, de tipo `int list -> int`, que retorne quantas vezes o elemento n de uma lista de inteiros `L` é maior que o elemento $n-1$. Você pode assumir a existência da função `toIncs` da questão anterior, caso não a tenha feito. Exemplo:

```
- incs (nil:int list);
exception match non exhaustive (* nao se preocupe com o erro. *)
- incs [1];
val it = 0 : int
- incs [1,2];
val it = 1 : int
- incs [1, 2, 1, 2, 2, 1, 2];
val it = 3 : int
```

Escreva seu programa aqui, um caractere por quadro: