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. São perguntas: "Posso fazer uma pergunta?"ou "Quanto tempo falta?"
- A prova termina uma hora e quarenta minutos após seu início.
- Seja honesto e lembre-se: **você deu sua palavra de honra**.

Alguns conselhos:

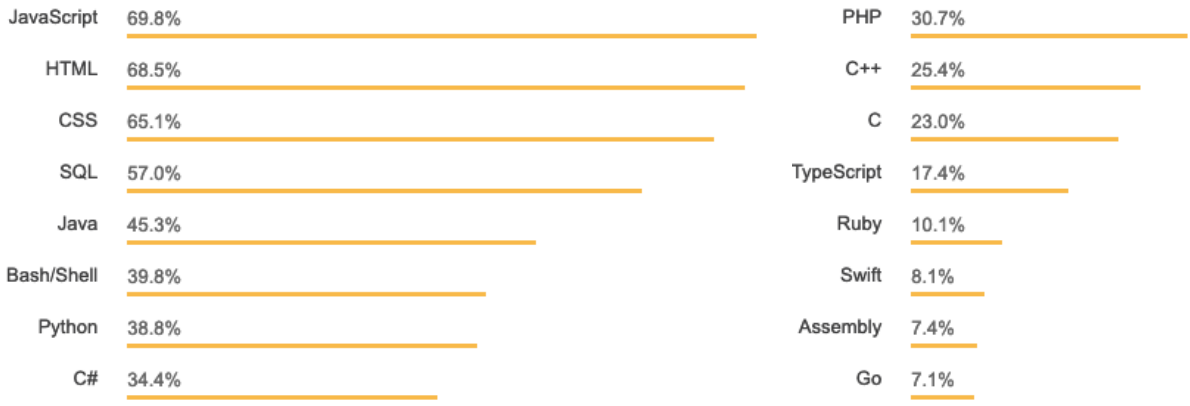
- Escreva sempre algo nas questões, a fim de ganhar algum crédito parcial.
- Se não entender a questão, e já tiver gasto sua pergunta, escreva a sua interpretação da questão junto à resposta.
- A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Questão 4	Extra

Questão Extra (0.5 Pontos): cite uma capital de algum país soberano (com assento na Assembléia das Nações Unidas) cujo nome comece pela letra 'F', em Português ou em Inglês.

1. (1 Ponto cada item) A figura abaixo mostra a classificação, por popularidade, das linguagens que aparecem no site de perguntas e respostas *StackOverflow*. Nesta questão você deverá escolher dez linguagens, e distribuí-las entre cada um dos dez itens abaixo. A linguagem atribuída a um certo item deverá apresentar a funcionalidade ou característica descrita naquele item. Importante: você não pode repetir uma linguagem em itens diferentes. Caso haja repetição, os itens em que a linguagem repetida aparece não serão considerados para fins de pontuação desta questão.



(a) Tipagem dinâmica:

(b) Tipagem estática:

(c) Tipagem fraca:

(d) Tipagem forte:

(e) Execução via compilação:

(f) Execução via máquina virtual:

(g) Escopo dinâmico:

(h) Escopo estático:

(i) Suporte a *closures*:

(j) Turing **in**completa:

2. A figura contém três exemplos de macros implementadas em C. Cada macro ilustra uma idéia diferente de alguma linguagem de programação. Em cada caso, indique que idéia ou característica é essa, que a macro emula em C, e cite uma linguagem de programação que a contém.
Importante: as respostas dos itens (a), (c) e (e) devem usar no máximo 25 caracteres.

```
#define SWAP(T, A, B) \
  {T _aux = A; \
   A = B; \
   B = _aux; }

void use_SWAP() {
  int x = 1, y = 2;
  double a = 3.1, b = 2.7;
  SWAP(int, x, y);
  SWAP(double, a, b);
}
```

(a - 1 Ponto) Característica:

(b - 1 Ponto) Linguagem:

```
#define ADD_A(x) x+a

void add_1(int *x) {
  const int a = 1;
  *x = ADD_A(*x);
}

void add_2(int *x) {
  const int a = 2;
  *x = ADD_A(*x);
}

void use_ADD_A(int x) {
  add_1(&x);
  add_2(&x);
}
```

(c - 1 Ponto) Característica:

(d - 1 Ponto) Linguagem:

```
#define TWICE(F,X) F(F(X))

int sqr(int x) {
  return x*x;
}

int inc(int x) {
  return x+1;
}

int use_TWICE(int p) {
  int x = TWICE(inc, p);
  int y = TWICE(sqr, p);
  return x + y;
}
```

(e - 1 Ponto) Característica:

(f - 1 Ponto) Linguagem:

(g - 4 Pontos) Caso todas as funções na linguagem de programação C fossem macros, como seria possível simplificar os registros de ativação dessa linguagem?

3. Nesta questão você deverá implementar um algoritmo de ordenação em SML de duas formas diferentes, para tanto, usando uma função `ins`.

   (a) (4 Pontos) Implemente uma função `ins`, cujo tipo deve ser `int * int list -> int list`. Essa função recebe um elemento e uma lista (supostamente ordenada), e insere o elemento na lista, produzindo uma nova lista, também ordenada. Exemplos:

```
- ins(3, [1, 2, 4]);
val it = [1,2,3,4] : int list
- ins(3, [1, 2, 3, 4]);
val it = [1,2,3,3,4] : int list
- ins(0, [1, 2, 3, 4]);
val it = [0,1,2,3,4] : int list
- ins(5, [1, 2, 3, 4]);
val it = [1,2,3,4,5] : int list
```

   Importante: sua resposta deve conter uma ou duas linhas.

   (b) (3 Pontos) Implemente uma função recursiva `inSortR`, de tipo `int list -> int list`, que utiliza a função `ins`, escrita na questão anterior, para order uma lista de números inteiros. **Importante:** sua função não pode utilizar `foldr` ou `foldl`. Caso não tenha feito a questão anterior, assuma a existência da função `ins`. Exemplos:

```
- inSortR [3, 4, 2, 6];
val it = [2,3,4,6] : int list
- inSortR nil;
val it = nil : int list
```

   (c) (3 Pontos) Implemente uma função `inSortF`, de tipo `int list -> int list`, que utiliza `foldr` ou `foldl`, mais a função `ins` do item (a) desta questão, para ordenar uma lista de números inteiros. **Importante**: sua função não deve ser recursiva –ela deve simplesmente invocar a função `foldr` ou `foldl`. Para ajudar-lhes com esse exercício, o início da resposta segue abaixo. Note que usamos `foldl`, mas `foldr` funcionaria também:

```
fun inSortF L = foldl ..........................................................
```

(d) Esta questão versa sobre *variáveis livres* no contexto do cálculo lambda. Para a solução da questão, considere o tipo abstrato `expr` abaixo, implementado em SML, que representa expressões lambda:

```
datatype expr = VAR of string | LAMBDA of string * expr | APP of expr * expr
```

i. (2 Pontos) Quais são as variáveis livres na expressão $\lambda x.y(\lambda y.y)xz$?

ii. (2 Pontos) Qual a forma normal da expressão $(\lambda x.\lambda y.xy)y$?

iii. (2 Pontos) Ré-escreva o termo:

```
val x = APP ((LAMBDA ("x", VAR "x")), LAMBDA ("y", APP (VAR "y", VAR "y")))
```

usando a sintaxe do cálculo lambda, i.e., $\lambda a.b\ldots$

iv. (4 Pontos) Escreva uma função `freevar`, que produz uma lista com as variáveis livres em uma expressão lambda. O tipo dessa função deve ser `expr -> string list`. Por exemplo:

```
- freevar (LAMBDA ("x", APP(VAR "x", VAR "x")));
val it = [] : string list
- freevar (LAMBDA ("x", APP(VAR "w", VAR "x")));
val it = ["w"] : string list
```

A fim de lhe ajudar a resolver esta questão, disponibilizamos uma função `diff`, de tipo `''a -> ''a list -> ''a list`, que remove um elemento de uma lista. A implementação de `diff` segue abaixo:

```
fun diff _ nil = nil
  | diff s (h::t) = if s = h then diff s t else h :: diff s t
```