Segunda Prova de Linguagens de Programação
- DCC024 -
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?"
- Você pode ir ao banheiro, mas deve deixar seu telefone celular sobre a mesa quando o fizer (e lembre-se do trato feito para esta prova).
- 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	Extra

Questão Extra (0.5 Pontos): O que poderia demandar o joelho de um menino que cresceu? A exemplo do que pediu o *Joelho Juvenal*? Que o menino fizesse furinhos na calça, para ele poder enxergar

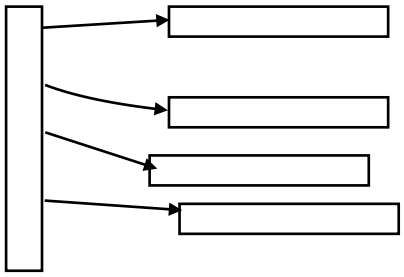
1. Esta questão refere-se à forma como arranjos são implementados em linguagens de programação imperativas.

   (a) (3 Pontos) Linguagens imperativas implementam arranjos de forma que seja possível termos acesso a qualquer posição em tempo constante; isto é, em $O(1)$. Como isso é possível?

Por que memória é contígua e o tamanho de cada célula é conhecido. Então o acesso acontece via base e offset

   (b) (3 Pontos) Como seria armazenado em memória uma estrutura como aquela apontada pelo ponteiro `p` no programa abaixo?

```
void init(int** p, unsigned N) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j <= i; j++) {
            p[i][j] = 1;
        }
    }
}
```



   (c) (1 Ponto) Qual dos dois programas abaixo deve ser mais eficiente?

Programa 1:

```
01 int sum = 0;
02 for (i = 0; i < M; i++) {
03     for (j = 0; j < N; j++) {
04         sum += m[i][j];
05     }
06 }
```

Programa 2:

```
01 int sum = 0;
02 for (j = 0; j < N; j++) {
03     for (i = 0; i < M; i++) {
04         sum += m[i][j];
05     }
06 }
```

O da esquerda

   (d) (3 Pontos) Justifique a sua resposta para a questão anterior.

C armazena arranjos por linha. Então o programa da esquerda possui melhor localidade de referência.

2. As questões a seguir pedem programas que devem ser implementados na linguagem de programação Prolog.

   (a) (3 Pontos) Implemente um predicado `sublist(L, S)` que seja verdade quando `S` for uma sublista de `L`. Por exemplo:

   ```
   ?- findall(L, sublist([a, b, c], L), R).
   R = [[a, b, c], [a, b], [a, c], [a], [b, c], [b], [c], []].
   ```

   ```
   sublist([], []).
   sublist([H|T], [H|L]) :- sublist(T, L).
   sublist([_|T], L) :- sublist(T, L).
   ```

   (b) (3 Pontos) Implemente um predicado `perm(L, P)` que seja verdade quando `P` for uma permutação dos elementos da lista `L`. Por exemplo:

   ```
   ?- findall(P, perm([a, b, c], P), R).
   R = [[a, b, c], [a, c, b], [b, a, c], [b, c, a], [c, a, b], [c, b, a]].
   ```

   Você pode usar o predicado `select`. Por exemplo:

   ```
   ?- findall((E, S), select(E, [a, b, c], S), R).
   R = [(a, [b, c]), (b, [a, c]), (c, [a, b])].
   ```

   ```
   nperm([], []).
   nperm([H|T], L) :- nperm(T, Lx), select(H, L, Lx).
   ```

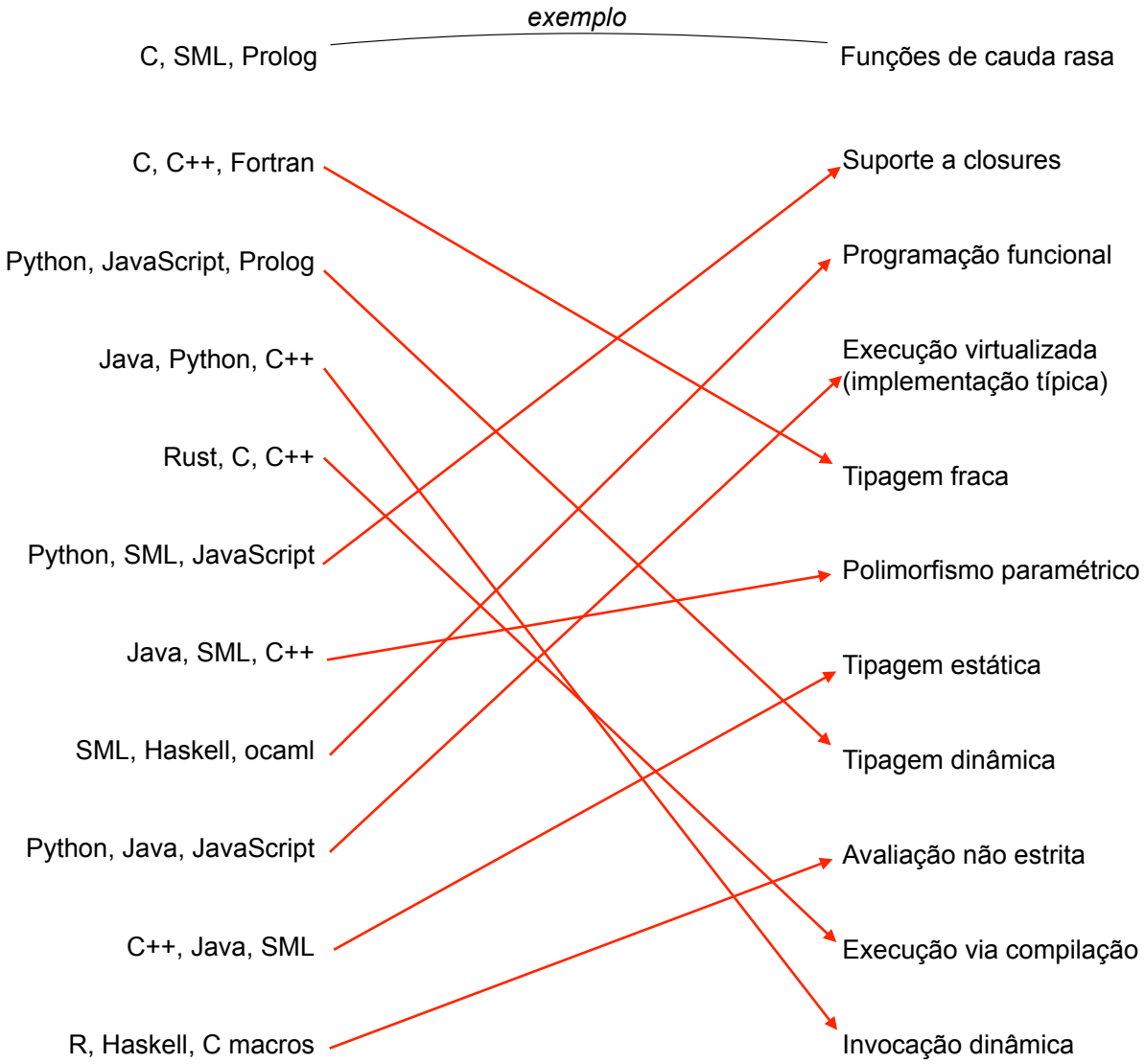
   ```
   perm([], []).
   perm(T, [H|LL]) :- select(H, T, L1), perm(L1, LL).
   ```

   (c) (4 Pontos) Escreva um predicado `anagram(L, A)`, que seja verdade se a lista `A` for um anagrama de alguma subsequência da lista `L`. Por exemplo:

   ```
   ?- anagram([l, i, v, r, e, s], [s, e, r, v, i, l]).
   true.
   ?- anagram([l, i, v, r, e, s], [s, a, r, v, i, l]).
   false.
   ?- anagram([l, i, v, r, e, s], [s, i, r, e]).
   true.
   ```

   ```
   anagram(L, A) :- sublist(L, S), perm(S, A).
   ```

3. Ligue as linguagens de programação à esquerda com características que elas possuem à direita. Você deve estabelecer uma bijeção entre as duas colunas. Cada resposta correta vale um ponto.



Era uma vez um joelho que se chamava Juvenal. Juvenal tinha um problema, coitado: vivia todo escalavrado. Também, quem mandou o Juvenal ser o joelho de um menino levado? Juvenal queria muito aprender língua de menino só pra dizer assim: "Menino, tem dó de mim!" Mas, quando o esfolado sarava, Juvenal bem que gostava de correr e de saltar. E ele se desdobrava e se dobrava outra vez todo alegre, pois sabia que, indo e vindo, fazia o menino feliz. (**Ziraldo**)