Lista de Linguagens de Programação – 25

Nome: _____ Matrícula: _____

1. A história da ciência da computação contém algumas linguagens de programação pioneiras, no sentido de que elas introduziram (ou popularizaram) conceitos e filosofias de programação que, embora óbvios hoje, não eram de maneira alguma definitivos quando surgiram. Cada uma das linguagens abaixo foi criada por um cientista da computação que veio a ganhar o prêmio *Turing*, a mais cobiçada homenagem em nosso campo. Cite uma das inovações que cada uma destas linguagens trouxe:

   (a) Fortran, criada por John Backus, IBM, 1954.

   (b) Lisp, criada por John McCarthy, MIT, 1957.

   (c) Simula 67, criada por Kristen Nyggard e Ole-Johan, Noruega, 1967.

   (d) ML, criada por Robin Milner, Escócia, 1974.

2. Algumas linguagens de programação são bem pequenas, outras são muito grandes. SML e Prolog são exemplos de linguagens pequenas. C++ e Perl são exemplos de linguagens grandes.

   (a) Em sua opinião, como se "mede" o tamanho de uma linguagem de programação?

   (b) Alguns projetistas de linguagens acham que uma linguagem de programação deveria prover em sua sintaxe básica tantos recursos quanto possível. Qual a vantagem desse tipo de cobertura?

   (c) Outros projetistas entendem que as linguagens deveriam ser criadas pequenas, e deveriam crescer conforme a necessidade de seus usuários. Quais as vantagens dessa abordagem?

   (d) Como é possível "crescer" uma linguagem de programação?

3. Java é um compromisso: a linguagem não é pequena. Por outro lado, ela não é tão grande quanto C++, por exemplo. Os projetistas de Java decidiram manter uma grande parte de suas funcionalidades em *bibliotecas*. Assim, embora arranjos em Java sejam imutáveis, vetores não o são, por exemplo. Cite uma vantagem e uma desvantagem de "crescer" uma linguagem via bibliotecas.

4. Nós não vimos muito de entrada e saída em Prolog. Este exercício vai suprir um pouquinho desta deficiência. Vamos fazer um programa para descobrir se uma palavra é um anagrama de alguma palavra válida de Português. Duas *strings*, s_1 e s_2 são anagramas se s_1 possui todas, e somente, as letras de s_2, possivelmente em alguma ordem diferente. O objetivo deste exercício é escrever um predicado `findAnagram(Dic, A, S)`, que seja verdadeiro se a *string* `A` for um anagrama de alguma palavra `S` no arquivo dicionário `Dic`. Por exemplo, se o nosso arquivo dicionário chamar-se `file.txt`, e possuir o seguinte conteúdo:

   ```
   baronesa.
   rosa.
   bena.
   fernando.
   ```

   Então a seguinte chamada deve acontecer:

   ```
   ?- findAnagram('file.txt', "roas", S).

   S = rosa ? ;
   ```

   Para realizar este exercício, você precisará escrever um predicado `file2list(FILE, LIST)`. Este deve ser capaz de ler um arquivo, `FILE`, tal qual `file.txt`, contendo uma palavra por linha, sendo cada linha finalizada com um ponto. O predicado então insere estas palavras em uma lista de *strings*. Dica: dê uma olhada nos predicados de entrada e saída: `see`, `inquire`, `seen` e `read`.

5. Já que a nossa prova está chegando, é hora de relembrar um pouquinho de ML. Neste exercício nós encontraremos a *mediana* de uma lista. A mediana é o "valor do meio", ou seja, quando a lista L de tamanho $|L|$ estiver ordenada, a mediana é o valor no índice $\lceil |L|/2 \rceil$. Um jeito fácil de resolver este exercício é ordenando a lista. Neste caso, podemos encontrar a mediana em $O(n \ln n)$ passos. Contudo, todavia e entretanto, a fim de tornar este exercício mais bacana, vamos ver quem são os homens (e eventuais mulheres) e quem são os ratos. Ratos ficam no $O(n \ln n)$, mas nós que somos bons na coisa, resolvemos tudo em $O(n)$. Você – que não é um rato – vai ter de achar a mediana em $O(n)$ passos.