Programando em LISP

Nesta atividade, vamos programar diversas funções LISP. Talvez algumas não tenham muito a ver com IA, mas devemos aproveitar esta ocasião para desenvolver habilidade em um paradigma alternativo (e muito menos sujeito a erros do que os convencionais).

A sala deverá se dividir em nn equipes, e cada equipe deverá desenvolver uma parte do aplicativo. A avaliação do trabalho será por inteiro. Ou seja, antes de entregar o trabalho as equipes deverão integrar os componentes e entregar um único programa. Ou todos tiram uma nota boa ou todos sucumbem.

Equipe 1

Este componente deve receber um arquivo tipo TXT de nome ENTRADA (possivelmente na forma de uma lista) e também receber um outro arquivo tipo TXT, de nome STOPWORD contendo as palavras não significativas e a partir destas duas entradas, deve elaborar um parecer quanto ao domínio e abrangência do idioma português do texto que está em ENTRADA.

Para este parecer, calcular:

1. X_1 = Quantidade de palavras diferentes (e que não sejam stopwords) utilizadas
2. X_2 = Quantidade de stopwords utilizadas no texto
3. X_3 = Quantidade de palavras distintas no texto
4. X_4 = Quociente de abrangência ($X_1 \div X_3$).

Equipe 2

Este componente deve ler um arquivo TXT, chamado ENTRADA e um dicionário de palavras corretas, denominada DICIONA, e devolva uma nova versão de ENTRADA, com o nome de SAIDA, no qual as palavras de ENTRADA que não estão em DICIONA, devem ser marcadas com um asterísco antes e depois da palavra inexistente.

Equipe 3

Este componente deve ler um arquivo TXT de nome ENTRADA e separe as palavras em categorias gramaticais. Após a separação, o programa deve imprimir as diversas categorias. Pode-se criar uma categoria gramatical denominada POÇO, que poderá conter as palavras que não se saiba separar.

Equipe 4

Este componente deve ler um arquivo TXT de nome ENTRADA e devolver o mesmo arquivo com o nome de SAIDA, contendo a transcrição da entrada em caracteres apropriados à pesquisa fonética. Usar-se-ão as seguintes regras:

Todas as palavras deverão passar pelo processo de fonetização. Este processo é uma busca e substituição de conjuntos de letras. A tabela de substituições deve ser externa, para poder ser atualizada de maneira independente, e deve começar com as seguintes ocorrências

Letra(s) original(is)	nova(s) letra(s)
quaisquer letras dobradas, exceto SS e RR	uma única ocorrência
SCH	X
CH	X
ASA, ESE, ISI, OSO, USU	AZA, EZE, IZI, OZO, UZU
CA, CO, CU	KA, KO, KU
SSE, SSI	CE, CI
QUE, QUI	KE, KI
consoantes mudas, ao final de palavras	excluí-las
Y	I
OU	O
IE	I
W	V
HA, HE, HI, HO, HU	A, E, I, O, U

Você deve esquentar um pouco a cuca para bolar este algoritmo. Sugiro criar a tabela completa de maneiras que embora a tabela fique grande, o processamento seja simples. Talvez haja necessidade de criar um caractere especial para representar o branco. Pense nisso (por exemplo para parametrizar uma constante no final da palavra).

Equipe 5

Esta equipe deve receber um arquivo de texto, no formato TXT, de nome ENTRADA e um segundo arquivo contendo duplas de palavras, denominado SINÔNIMO. O programa deve fazer duas tarefas:

a) enriquecer o texto, fazendo uma substituição de palavras, sempre que estas contarem com um sinônimo no segundo arquivo. Por exemplo, se tivermos no arquivo SINONIMO

```
casa residencia
casa moradia
casa tugurio
```

Se o texto de entrada for *a minha casa é bonita, a minha casa tem janelas e a minha casa é verde* deve ficar *a minha casa é bonita, a minha residência tem janelas e a minha moradia é verde*. Desprezar a preocupação com a concordância.

b) Empobrecer o texto, trocando todas as ocorrências de sinônimos pela mesma palavra.