Modulo 3 – calcolo del "fattore di SPAM" di una email – bozza 3

In questo modulo si deve realizzare il dizionario in cui sono memorizzati i pesi delle parole delle email, con i quali si deve calcolare il "fattore di SPAM" di una email, ovvero la probabilità che essa sia SPAM piuttosto che HAM. A questo scopo si deve implementare la struttura dati sotto forma o di hash-table oppure di dizionario a vostro piacere. Deve essere inoltre realizzata la funzione che legge le coppie <token-valore> da un file e le inserisce nel dizionario.

*struct Dizionario * nuovoDizionario(int dimensioni);*

Crea un nuovo dizionario (o hashtable) contenente un numero massimo **dimensioni** di elementi. Se si ha errore si torna NULL. Definite VOI la struttura dati **struct Dizionario**, che consegnerete nel file dizionario.h tramite la pagina web di consegna.

*int freeDizionario(struct Dizionario * dizionario);*

Dealloca tutta la struttura dati **dizionario**. Torna i soliti codici risultato.

*int inserisciValore(char * chiave, double valore, struct Dizionario * diz);*

La funzione inserisce nel dizionario **diz** un nuovo **valore** in corrispondenza della stringa **chiave**. Se necessario si torni un opportuno codice di errore, ad esempio se la tabella è piena si torni TABLE_FULL e se la chiave è già presente si torni DOUBLE_KEY_FOUND.

*int cercaValore(char * chiave, struct Dizionario * diz, double * valore);*

La funzione cerca nella tabella **diz** la chiave **chiave** e, se presente, ne torna il valore associato nella variabile (passata per puntatore) **valore**. Se l'elemento non è presente si torna il codice di errore KEY_NOT_FOUND. Per altri codici di errore si vedano i soliti file .h .

*int leggiTokens(char * filename, struct Dizionario * diz);*

La funzione legge un file di testo di nome **filename** in cui sono presenti solo righe contenenti, ciascuna, una coppia "Token valore", in cui il token è nella forma stampata da stampaTokens del modulo1 (ad esempio "From*andrea"), seguito da spazio e poi da un numero in virgola mobile (un double). Per ciascuna riga letta si inserisce nel dizionario, usando come chiave il token, il valore letto. Si tornino opportuni codici di errore se necessario.Si assuma che il file sia corretto e che finisca per '\n'.

Esempio di contenuto della riga: "From:*Andrea 0.9\n"

*int calcolaPeso(Token tokens[], int quanti, struct Dizionario * diz, double * peso);*

Si supponga di aver tokenizzato una email e di aver ottenuto il vettore di **quanti** Tokens **tokens** (anche ripetuti).
Si vuol calcolare il peso della email (ovvero la probabilità che si tratti di SPAM) usando la procedura che segue:

- per ciascuno dei tokens contenuti nel vettore **tokens** si trovi il corrispondente peso W_i in **diz**.
  - se il token non è presente si consideri che il suo peso W_i è 0.4 (leggermente HAM)
- si cerchino i primi 20 token **diversi** (tra quelli elencati) il cui peso W_i dista maggiormente dal valore 0.5 (ovvero quelli che sono forti indicatori di SPAM o forti indicatori di HAM)
  - ovvero si ordinano i token per distanza decrescente e si contano i primi 20 **diversi**
  - i token che hanno la stessa distanza da 0.5 che il ha il ventesimo vanno sommati anche loro
  - ovvero ci si ferma al primo token che dista da 0.5 di meno del ventesimo
- usando i pesi selezionati si calcolino i due prodotti **Bad** $= \prod W_i$ e **Good** $= \prod (1-W_i)$
- il peso della email è pari a: **Bad/(Bad+Good)**

Funzioni per il test

Come al solito, si scrivano le funzioni di test definite nel file testModulo3.h