Middle East Technical University
Department of Computer Engineering

CENG 140

Spring '08

Take Home Exam 1

REGULATIONS

Due date: 16 April 2008, Wednesday *(Not subject to postpone)*

Submission: Electronically. You will be submitting your program source code written in a file which you will name as `hw1.c`

through the cow web system. Resubmission is allowed (till the last moment of the due date), The last will replace the previous.

Team: There is **no** teaming up. The take home exam has to be done/turned in individually.

Cheating: All parts involved (source(s) and receiver(s)) get zero.

INTRODUCTION

This problem is about identifying a text language. A human approach to this problem could be an attempt to spot well known and frequently used words form languages. So a use of 'the' would imply 'english', a use of 'avec' would imply 'french' etc.

Our approach will be totally different. We will be making use of the computers ability to process data in huge amount. The concept that we will employ is called 'bigrams'. In general n-gram is the grouping of n many entities in a neighborhood. A bigram is the neighborhood of two entities. In our case 'entities' are letters and some punctuation characters and neighborhood means "being followed by" (in a word). So for example in the word "`ATTRIBUTED`" (considering also the surrounding spaces as letters '␣') Has the following bigrams:

␣A, AT, TT, TR, RI, IB, BU, UT, TE, ED, D␣

The frequency of bigrams differ from language to language. So bigram frequency can be considered as a signature of the language. Of course to be conclusive the amount of the text analyzed must be large. Just as an example (taken from www.cryptograms.org/letter-frequencies.php) The most often occurring bigrams and their frequency percentage (%) in English is

TH	3.88	HE	3.68	IN	2.28
ER	2.18	AN	2.14	RE	1.75
ND	1.57	ON	1.42	EN	1.38
AT	1.34	OU	1.29	ED	1.28
HA	1.27	TO	1.17	OR	1.15
IT	1.13	IS	1.11	HI	1.09
ES	1.09	NG	1.05		

PROBLEM

In this work you are expected to write a program that reads form the standard input and write to standard output. You will read a text, character by character, which is a number of corpi (corpi is the plural of corpus. corpus means a 'body of text'). Each corpus is a text in a language. There is no limitation on the size of the corpus. A corpus is made of a sequence of words separated by single spaces. Words are made of uppercase letters from the English alphabet. How corpi are separated from each other is detailed in the next section (SPECIFICATIONS). You will count all

bigrams and find the percentage of occurrence. Then, again form the input you will be supplied with an unknown language where you will identify which language it is most similar to. For this purpose you need a measure by which you can identify the most similar language. Here we provide you with a metric of dissimilarity:

$$dissimilarity(language_1, language_2) = \sum_{\xi \in \text{all bigrams}} |\Omega(\xi, language_1) - \Omega(\xi, language_2)|$$

where $\Omega(\xi, language)$ is the percentage of occurrence of the bigram ξ in the *language*.

So, as much as the $dissimilarity(language_1, language_2)$ value is close to zero, we can conclude that these two languages are similar.

SPECIFICATIONS

- There will be at most corpi for 9 languages (except the unknown language).
- Corpi will be separated from each other by a line of the form
  `#`n
  where n will be successive integers starting with `1`. So, on top of the first corpus is a line
  `#1`
  then comes the first corpus (which may contain whitespaces). Then there will be a line
  `#2`
  marking the start of the second corpus. The second corpus will follow, and so on.
- Input starts with `#1`, continues with the input of all corpi and finishes with a separate line of a single question mark
  `?`
  which is followed with the corpus of the unknown language.
- There can be arbitrary many whitespaces separating words. Consecutive white spaces are equivalent to a single space '␣'.
- In the bigram computation consider '␣' (space) as a letter.
- For bigram counting use `long int` as type. For percentages use `float`.
- The output is the number of the language you have determined to be the most similar to the unknown language. (Yes, it is correct; you will be outputting a single digit number only!)
- You are <u>not</u> allowed to use functions from `string.h`. Worse then that, you are not allowed to create strings or make use the string concept. (You actually do not need strings at all).
- You <u>cannot</u> use pointer variables; the dereference operator. The only allowed 'pointer of' operator is the one used in `scanf` calls.
- You are advised to device a two (actually three: after reading bullet see below) dimensional array of `long` elements of size $m \times m$ where m is the cardinality of the alphabet. Letters' ASCII codes can be used (subtracting some offset) to index the array. So, `array['T'-`λ`]['E'-`λ`]` would hold the count of the bigram `TE`. What shall λ be? How can we deal with the space character? Think about it. You have to solve these issues.
- Since you will have more than one corpus it is wise to device a three dimensional array, indeed, the first index which runs over the language number (so is actually of the size `[10]`). The second and the third indices are as explained in the item above (and are both of size m).