syllabus for

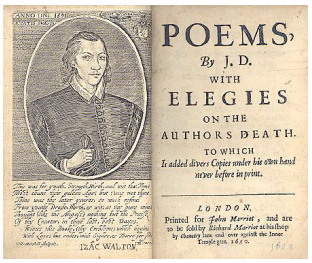
computing for poets

comp 131

Instructor: **Mark LeBlanc** SC-103 508.286.3970
mleblanc@wheatoncollege.edu Hours: M 2-3:30, R 9:30-11
http://cs.wheatoncollege.edu/mleblanc

Meeting Times: Tuesday, Thursday 11:00-12:20
Room A102, Science Center

Come live with me, and be my love,
And we will some new pleasures prove
Of golden sands, and crystal brooks,
With silken lines and silver hooks.
John Donne, The Bait



The use of computers to manage the storage and retrieval of written texts creates new opportunities for scholars of ancient and other written works. Recent advances in computer software, hypertext, and database methodologies have made it possible to ask novel questions about a poem, a story, a trilogy, or anthology. This course teaches computer programming as a vehicle to explore poems and other texts that are now available online. Programming facilitates top-down thinking and practice with real-world problem solving skills such as problem decomposition and algorithmic thinking. Programming on texts introduces students to rich new areas of scholarship including stylometry and authorship attribution. Prerequisites: A love of the written (and digital) word; no computer programming experience required.

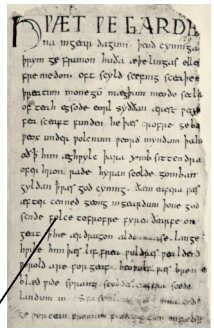
Using computers to analyze poems and stories is an exciting new area of research. Although many tools exist for working *with* texts, what do you do when the tools you have at your disposable cannot answer your questions? In this course, you will learn to write computer programs (also called "software" or "scripts") that can answer your original questions. The programming language we will learn is named Perl (Practical extraction and reporting language). Perl is a powerful language when you are dealing with strings of characters. Some of the programs that you will write this semester will analyze texts to:

- compute the percentage of vowels used in a text;
- search for patterns of letters or words using the powerful pattern matching language of regular expressions; for example, how many six letter palindromes can you find?
- search your own papers that you have written in the past for questionable writing style;
- determine the top-10 most frequently used words by an author;
- find *hapax legomena* (words that appear only once in an entire collection of works)
- build a concordance of all the words in a poem or the entire Anglo-Saxon corpus;
- learn the beginning steps in authorship attribution as you keep statistics for the relative frequencies of the most commonly used words in a poem, story, or corpus
- compare a sample work from one author with those from other authors

CONNECTION:

Computing and Texts
Eng 208 Anglo-Saxon Literature or Eng 259 J.R.R. Tolkien
with Comp 131 Computing for Poets

This course is "connected" with two courses from Professor Mike Drout in English: Anglo-Saxon Literature and J.R.R. Tolkien. The relationship between Old English poems has been a vexed question for nearly 150 years. Most of the poetry is anonymous and exists only as tenth-century copies in manuscripts (some of it is assumed to be much earlier). We have only three named authors of poetry in the Anglo-Saxon period (Cædmon, Cynewulf and King Alfred), and there are various problems with linking these names (much less their biographies) with more than a very few specific poems. Although a few prose texts are by known authors (particularly the homilists Ælfric and Wulfstan), even the majority of the prose is anonymous. Thus for years scholars of Old English have struggled to divine relationships between texts based on vocabulary, meter, and style. These results have been at best contentious and at worst completely unsuccessful. As we proceed to learn more and more Perl in this course, we will set up and run experiments using the entire Anglo-Saxon corpus. Some of the questions you could ask may never have been asked before.



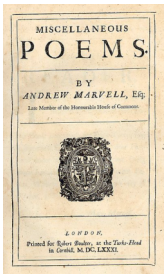
```
while ($nextWord = <AngloText>)
{
        $nextWord =~ tr/æ/AE0/;
        :
        :
```

We will learn to combine Perl programs, comma-separated output from those Perl programs, and Excel spreadsheets in a manner very similar to the way your instructor does when doing research. A major "take home story" for this course is to learn ways to view and analyze texts in an entirely new way. By leaning on the computer, new methods of analysis that you could not do by hand will now be at your disposal.

In addition to programming in Perl and working in Excel, we will also study how computers store individual characters, including the traditional (English-only) ASCII character code and the international standard called UniCode.

NOTE: This course is but an introduction to using computing to study written texts. Computers allow us to study texts in exciting new ways that we could not otherwise do; however, as we'll discuss at length, we are wise if we keep in mind what computers cannot do. The following quotes can help us (1) stay humble and (2) stay focused.

"As students of a powerful new form of scholarship, we have much to offer.
We do ourselves no justice when we forget that the quantifiable features
we deal in are but the shadow of a shadow."
John Burrows, Computers and the Humanities, v37, 2003, p30.


MISCELLANEOUS
POEMS
BY
ANDREW MARVELL, Esq;
LONDON,
Printed for Robert Boulter, at the Turks-Head
in Cornhill, M. DC. LXXXI.

"The onus of competency, clarity, and completeness is on the practitioner. The researcher must document and make clear every step of the way. No smoke and mirrors, no hocus-pocus, no 'trust me on this.' "
Joseph Rudman, Computers and the Humanities, v31, 1998, p353.

In computer science, if you are almost correct you are a liability.
Fred Kollett (1941-1997), MathCS, Wheaton College, Norton, MA

Text:

Perl for Exploring DNA by LeBlanc and Dyer
Oxford University Press 2007

What, a book about DNA? **This is a small book that teaches Perl programming** to new programmers who are dealing with lots of text, e.g., strings of characters. Yes the theme is on searching DNA, but DNA is a language of sorts. The book uses a linguistic metaphor throughout. In our case, the book could be re-titled: Perl for Exploring *Texts*.

I also strongly recommend that you buy a 3-ring binder. I'll pass out lots of handouts.

Your Grade:

Things to do	Grading Percents	Frequency
Labs	5% overall	in class as needed
Homeworks	15% overall	TBA
5 Programs	50% overall	
p1: Hello Poet!	4%	TBA
p2: Vowel Counter	8%	TBA
p3: How's My Writing?	8%	TBA
p4: Author Attribution (Part I)	10%	TBA
p5: Author Attribution (Part II)	20%	TBA
Exams	30% overall	
Exam I	15%	Thur, Feb. 28
Exam II	15%	Thur, Apr. 17
No Final Exam		

Late Submissions:
Due is due. Always turn in whatever you have on time. Something turned in on time is much better than not having it accepted because it is late. Late is not an option. (Good, glad we can all agree with this).

Honor Code Revisited:
It goes without saying that all submitted work will be the student's own, in keeping with the Wheaton Honor Code, unless the assignment has assigned groups. For labs, you may get "help" from fellow classmates, but remember that all completed work must be your own. Use discretion; don't ask your colleague for "the" answer or for lines of Perl. However, I do encourage you to discuss the problem in general, such as the type of statements or functions one might use. For homework, your answers and software must be your own from beginning to end. Here is an analogy. Almost no one would every "use/steal" a line or two from another person's poem. Consider it the same with your Perl programs. Don't "borrow/use" lines or sections of Perl from another classmate. Your program is (like) your poem; everyone's program should be unique. Be wise. If a colleague is asking you for too much help, be honest and remind them your program is just that, *your* program.

Roads Go Ever On and On
J.R.R. Tolkien

Roads go ever on and on,
Over rock and under tree,
By caves where sun has never shone,
By streams that never find the sea;
Over snow by winter sown,
And through the merry flowers of June,
Over grass and over stone,
And under mountains in the moon.

Tips for working on your own ....

(0) It is expected that you spend at least 2-3 hours on reading, study and preparation for every 90 minutes of lecture and discussion.
(1) It is expected that you spend at least 6-10 hours per week on your current programming assignment. WARNING: Programmers typically underestimate the time it takes to complete a software project; 6-10 hours per week on your programming assignment may be one of those "underestimations."

In classroom "LABS"

(0) The computer work in class (labs) are a critical part of the course. In a way, it is your time to "hack", solve unique problems, and show that you can work hard on the problem at hand. Your labs will prepare you to work on your next programming assignment. You must be in lab to get credit for the session. If you happen to miss a lab you are strongly encouraged to do it on your own time, but please do not ask for credit.
(1) In order to best grasp the material presented in the lab, I strongly suggest that you completely redo any labs that you find difficult. (Read that last sentence again, unless of course you've already reread it once).

Exams

Your two exams will test your comprehension of Perl as well as your ability to write your own Perl. During lecture, I will give "hints" of what a typical test question might be; take good notes! There will be no makeups, nor will the lowest exam be dropped. If you are an athlete and/or you have a conflict with an exam date, please see me within the first week of classes.

Homeworks

A few homeworks are sprinkled throughout the semester. These are mostly to provide you with deadlines, practice with time management, produce (next) drafts of your programs, and a chance to take a lab one step further.

HELP

I have listed my office hours on the syllabus. But I'm always near a keyboard so we can schedule a time to meet. Study, study, study and talk about it with me and others as often as you can.

Please don't wait too long before you see me;
a quick chat in my office can often clear things up.
I'm here a lot...

Metrical Charm 8: For a Swarm of Bees

SaxonWiþ ymbe nim eorþan, oferweorp mid þinre swiþran
handa under þinum swiþran fet, and cwet:
Fo ic under fot, funde ic hit.
Hwæt, eorðe mæg wiþ ealra wihta gehwilce

```
while (nextWord =~ m/swiþran/g )
{
     foundAnother++;
     :
     :
```