## Introduction to the Use of Computers

# Text editing and encoding

### Friday 19th October 2012

This lab session is about editing text, and the ways in which text is encoded for storage to a file system or for transmission to processes.

1. This part of the lab introduces you to a programmer's text editor, Emacs.

   (a) using Mac OS X as the operating system, start the Emacs application. It should be present in the dock (the series of icons at the bottom of the screen); if it is not, there should be an application launcher for Emacs in the Applications section of the Finder (the file manager).

   (b) Emacs comes with comprehensive built-in help, including an introductory tutorial. Firstly, launch the tutorial by issuing the following keystrokes: `C-h t` (meaning: first, hold down the control key and, while still holding it down, hit '`h`'; then, let go of the control key (and the '`h`') and hit `t`). The tutorial should replace the initial startup screen; reading carefully, follow the tutorial's instructions from beginning to end.

2. This part of the lab introduces some more of Emacs' functionality. Emacs comes with functionality not just for text editing but for all sorts of tasks which programmers and other computer professionals routinely perform. One such task is investigation and editing of binary files.

   (a) from the course website, download the three '`hello`' files used in lecture 1 to introduce the ASCII encoding.

   (b) open the simple text file `hello.txt` in Emacs (use `File→Visit New File` from the menu bar, or `C-x C-f`). Verify that the buffer's contents actually contain the text you expect.

   (c) activate the binary mode by typing the following keystrokes: `M-x hexl-mode RET` (That is, the `M-x` or `ESC x` extended command keystroke covered in the tutorial, followed by `hexl-mode` typed normally, followed by the return key). At this point you may be prompted about discarding undo information, in which case you should accept (by hitting '`y`'). The display of the file should change to one where (side-by-side) the file contents are displayed first in hexadecimal, then as characters. Investigate whether the ASCII encoding given in lecture 1 is correct.

   (d) exit the binary mode by typing `M-x hexl-mode-exit RET`; again, you may be prompted about discarding undo information, in which case you should accept (by hitting '`y`').

   (e) repeat the previous steps with the html file referred to in lecture 1. When in the normal (text-viewing) mode, you should be able to type `C-c C-v` to display the html in a web browser.

   (f) repeat the previous steps with the postscript file referred to in lecture 1. When in the normal (text-viewing) mode, you should be able to type `C-c C-c` to display the rendered postscript directly within Emacs.

3. This part of the lab is a series of questions to reinforce the material covered in lectures. If you wish, you may use Emacs (introduced in the previous part) or other text editors or similar tools to help you answer the questions.

   (a) For each of the following statements, state whether it is true, false or arguable (and if ‘arguable’, explain why):

      i. ASCII is a character set;
      ii. ASCII is a character encoding system;
      iii. ASCII is a 256-character repertoire;
      iv. ASCII is normally encoded in 7-bit bytes;
      v. ASCII is a fixed-width encoding;
      vi. ISO-8859 repertoires allow multiple-language texts;
      vii. ISO-8859-1 is a 256-character repertoire;
      viii. ISO-8859-1 is a good solution for Western European languages;
      ix. the string `£10` is encoded in ISO-8859-1 as `a3 32 31`;
      x. ISO-8859-1 is a fixed-width encoding;
      xi. Unicode is a character set;
      xii. Unicode is a character encoding system;
      xiii. Unicode covers all known natural languages;
      xiv. Unicode allows multiple-language texts;
      xv. UCS-2 is a fixed-width encoding;
      xvi. UCS-2 allows encoding of texts made from arbitrary Unicode characters;
      xvii. UTF-16 is a fixed-width encoding;
      xviii. UTF-16 allows encoding of texts made from arbitrary Unicode characters;
      xix. the string `£10` is encoded in UTF-16 as `00 a3 00 31 00 30`;
      xx. UCS-4 is a fixed-width encoding;
      xxi. UCS-4 allows encoding of texts made from arbitrary Unicode characters;
      xxii. UTF-8 is a fixed-width encoding;
      xxiii. UTF-8 allows encoding of texts made from arbitrary Unicode characters;
      xxiv. the string `£10` is encoded in UTF-8 as `c2 a3 31 30`;