# TP6

The course homepage is here:

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1415/`.

You will find the slides from the course and some other files for the exercise there.
Details of shell commands and C functions can be obtained by using the `man` command.

## 1 Character encodings

As we saw in the course, there exist different ways to represent characters. A *character set* is a mapping of integers (also called *code points*) to characters (letters, digits, punctuation marks etc). The most important character sets that one encounters in a Western European context are:

- ASCII, whose domain is 0..127;
- the so-called Latin-1 (ISO 8859-1) extension of ASCII, covering the domain 128..255;
- Unicode, compatible with ASCII/Latin-1, but defining a much larger code space (hex 0..1FFFF).

A *character encoding* describes how to describe a code point (or more generally, a sequence of them). For ASCII/Latin-1, the encoding is trivial, each byte describes one code point. For Unicode, one uses a variable-length encoding called UTF-8 (which was discussed in the course). In this encoding, a code point is represented by 1 to 4 bytes.

1. Take the program `unicode.c` and make it output the following city names correctly.

   Charleville-Mézières (France)
   L'Haÿ-les-Roses (France)
   Kroměříž (Czechia)
   Gödöllő (Hungary)

   This requires two sub-tasks:

   - Find out the Unicode codepoints for the non-ASCII characters in the names above, such as ÿ ("y with diaresis") etc. Change the symbolic constants in the beginning of `unicode.c` accordingly.
   - Complete the function `utf8` that takes a Unicode code point and outputs its UTF-8 representation.

2. Among the files that you find on the course web page, there is an HTML file that does not display correctly (`tcheque.html`). What went wrong? How can we repair it?

3. Write a program that repairs the broken HTML file, based on the partially completed program `repair.c`.

## 2 Gray code

An $n$-bit Gray code is a sequence of $2n + 1$ bit patterns of length $s$, starting and ending at $00\cdots00$, that visits every other $n$-bit pattern and changes only one bit between two successive patterns. For instance, for $n = 3$:

000, 001, 011, 010, 110, 111, 101, 100, 000

On the course page you will find a skeleton program (`gray.c`) that should output the sequence for a given $n$. Your task is to complete it. Think before you act! If your code is longer than, say, 10 lines, then it is too long...

## 3 Barcodes

In the course, we have seen methods for error-detecting/correcting codes. We shall consider another code, called *two-out-of-five*, which is commonly used in barcodes. Here, five bits are used to represent a single decimal (0..9) digit. A valid code has two bits set (1) and three bits unset (0). Thus, there are exactly ten valid codes. A *weight distribution* assigns to each of the five positions a weight (from 0..9), and the value of a five-bit code is obtained by adding the weights of the bits that are one. Typically, 0 cannot be represented in this way.

1. How many different (modulo ordering) weight assignments are there that can represent all digits from 1..9 (one of which will be represented twice)?

2. Clearly, all two-out-of-five codes can detect a single bit error. Can any of your codes also *correct* a single error?

3. One common standard for barcodes making use of the above encodings is the *interleaved 2-out-of-5* code (see also the link on the course webpage). The code uses bars (black) and spaces (white) that can be either narrow or white.

   - A barcode starts with a sequence *narrow bar, narrow space, narrow bar, narrow space* and terminates with a sequence *wide bar, narrow space, narrow bar.*
   - In between, pairs of digits $(c, d)$ are encoded by interleaved bars and spaces, i.e. $c$ by five bars and $d$ by five spaces.
   - In a sequence of five bars or spaces, “wide” means 1 and ”narrow” means 0. The weight assignments are 1, 2, 4, 7, 0, in that order, where the digits 1..9 are encoded naturally, and the digit 0 is encoded as 4+7=11.

   The program `barcode.c` contains some infrastructure to generates these barcodes and display them graphically. It remains to complete the function `encode`, which is supposed to take two characters and produce their interleaved encoding.