

**Homework 1****due Monday 9 Feb.**

(1) Convert the numbers 8000, 7999, and 4095 (given in decimal) to hexadecimal, binary, and octal. **Show your method (you have to use one explicitly).** You can use Python functions `hex`, `eval`, `oct`, `int(string,base)` to check your answers.

(2) Convert the following to decimal: `0xABC1`, `0o7036`, `0b111001000`. **Show your method (you have to use one explicitly).**

(3) Write a Python (or C++) function `convert(N,b,c)` that converts the non-negative integer from base `b` representation to base `c` representation. Here, `N` is a **list** of base `b` digits. The function should return a list of base `c` digits.

The digits in a base `B` are the integers from 0 through `B - 1`. If you write in C++, dynamic arrays should be used instead of lists.

(4) Add the bit-strings, and find the final carry bit `C`.

(4a) `0 0100 1011 + 0 1101 0001`

(4b) `1 0001 1101 + 0 1110 1000`

(5) Let `n` be a non-negative integer.

(5a) How do you tell from the base `b` representation of `n` whether `n` is odd or even?

Case 1: `b` is odd. Case 2: `b` is even.

(5b) How can you tell from the base ten representation of `n` whether `n` is a multiple of nine?

(5c) How can you tell from the octal representation of `n` whether `n` is a multiple of nine?

(6a) Convert the character sequence `"x = .2"` into 7-bit ASCII. (The quotes are not part of the string.)

(6b) In base 3 count from zero through ten.

(7) Let `M` be an array (or list in Python) of integers with entries `3, 1, 0, 5, 0, 6, 0`,

(In C++, `int M[7] = {3, 1, 0, 5, 0, 6, 0}`; in Python, `M = [3, 1, 0, 5, 0, 6, 0]` )

(7.a) What is the value of each of the following? (When a reference is out of bounds, say `error`.)

`M[0]`, `M[M[0]]`, `M[M[M[6]]]`, `M[3]`, `M[M[0]+2]`, `M[M[0]]+2`, `M[M[5]+2]`

(7.b) Draw the array with its entries (in the array), with its indices (above the array entries), and with arrows (under the array) that represent pointers.

(7.c) We now execute the following sequence of commands. Show the output of each `print` statement (equivalent to `cout <<` in C++).

```
print M[M[2]]
print M[M[4]]
M[M[2]] ← 8      # Assignment
print M[M[2]]
print M[M[4]]
```

Why did *one* assignment change both `M[M[2]]` and `M[M[4]]` ?