Preamble

Welcome to the course! This worksheet is intended to act as a refresher for some of the basic computer science you will require for this course and is NOT representative of either the course or the real worksheets. You may find this new or trivial — please ensure that you are able to answer questions 1 to 3. Questions will be reviewed in class. You can hand in the answers to any or all that you wish at the hand-in area by the maths reception by Thursday.

Questions

1. Recall that digital computers store data in binary format, ie. that information is encoded as bit-strings, a collection of ordered *Bits*, each of which may take the value of either 0 or 1. A *Nibble* is 4 consecutive bits and a *Byte* is 8 consecutive bits. With bit-strings, bytes and nibbles, we index from the right-most, or *least-significant bit*, starting from 0.
   We will sometimes use the notation N_b to denote the number N is to be interpreted in base-b notation, ie. 11_2 is 3 in base-10 and 11_{16} is 17 in base-10.

   (a) How many bits does it take to represent the number $N \in \mathbb{N} \cup \{0\}$?

   Hexidecimal notation is often used for compact human readability of bytes. Each byte is represented by consecutive nibbles in hexadecimal format. In hexidecimal (base-16) format, the numbers $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$ are represented correspondingly with the symbols $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$,A,B,C,D,E,F. As each nibble consists of 4 bits, each nibble is represented by a single hexidecimal character and each byte by two consecutive nibbles. We will leave a space between bytes for readability and pad with zeros to represent full bytes.

   (b) Convert the number 14598366_{10} to hexadecimal format.

   (c) What is the base-10 representation of CA FE?

   Recall the XOR (exclusive-or) operation. If a and b are bits then $a \oplus b = 1$ if $a \neq b$ and 0 otherwise. The XOR operation may be extended to two bit-strings of arbitrary but equal length by applying it separately to bits of the same index, ie. $101 \oplus 111 = 010$.

   (a) What is the effect of XORing 0 with any bit? What is the effect of XORing 1 with any bit?

   (b) What is 0D AD $\oplus$ A1 10 in hexadecimal?

2. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is a *negigible function* if for every positive polynomial[1] $p(n)$ we have that there exists $N \in \mathbb{N}$ such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

   (a) Prove that the function $f : \mathbb{N} \to \mathbb{R}$ given by $f(n) = 2^{-n}$ is a negligible function.

   (b) Prove that if negl_1 and negl_2 are negligible functions then the function $\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ is also a negligible function.

   (c) Prove that for any positive polynomial p and any negligible function negl_1 we have that the function $\mathsf{negl}_4(n) := p(n) \cdot \mathsf{negl}_1(n)$ is also negligible.

[1] A function p is *positive* on a set S if has the property that $\forall s \in S$ we have that $p(x) > 0$.

3. Recall the *Big-Oh* notation. Let $S \subseteq \mathbb{R}$ and f, g be two positive functions defined on S with images in $\mathbb{R}$.
   We say that $f(n) = O(g(n))$ (or $f(n) \in O(g(n))$) if there exists $C \in \mathbb{R}_{>0}$, $N \in \mathbb{N}$ such that for all $n > N$ we have that $|f(n)| \leq C|g(n)|$.
   We say that $f(n) = o(g(n))$ (or $f(n) \in o(g(n))$) if for all $C \in \mathbb{R}_{>0}$ there exists $N \in \mathbb{N}$ such that for all $n > N$ we have that $|f(n)| \leq C|g(n)|$.
   We say that $f(n) = \Theta(g(n))$ (or $f(n) \in \Theta(g(n))$) if there exists $C_1, C_2 \in \mathbb{R}_{>0}, N \in \mathbb{N}$ such that for all $n > N$ we have that $C_1|g(n)| \leq |f(n)| \leq C_2|g(n)|$.

   (a) Consider the following functions and re-order them in increasing order of their asymptotic growth rates. Proofs are not required. Assume that $0 < a < 1 < b$.

   $$\ln n, \quad b^n, \quad \exp(\sqrt{\ln n \ln \ln n}), \quad n!, \quad b^{b^n}, \quad 1, \quad n^n, \quad \ln \ln n, \quad n^a, \quad n^b, \quad n^{\ln n}$$

   (b) L-notation is commonly used in characterising the behaviour of algorithms for the discrete logarithm and factoring problems. Consider where

   $$L_n(\alpha, c) = \exp\Big((c + o(1))(\log n)^\alpha (\log \log n)^{1-\alpha})\Big)$$

   where $c > 0$ and $0 \leq \alpha \leq 1$ fits into the ordering in part (a) when c and α are varied.

4. Modern software is both useful in learning and researching cryptography. Briefly investigate

   (a) *Cryptotool 2* — an open-source electronic learning toolkit for cryptography. Available either soon on the Oxford computer systems or from `https://www.cryptool.org/en/cryptool2`.

   (b) *SageMath* — an open-source computer-algebra system based upon the python programming language. SageMath will be used later in the course to implement several algorithms for cryptanalysis. Available either for download at `https://www.sagemath.org` or in the cloud at `https://sagemathcloud.com` which requires free registration. For now it is recommended that you simply explore the system using the cloud.