University of Waterloo
CS240E, Winter 2026
Written Assignment 1

Be sure to read the assignment guideliness (`http://www.student.cs.uwaterloo.ca/~cs240e/w26/guidelines.pdf`). Submit your solutions electronically to MarkUs as **individual** PDF files named a1q1.pdf, a1q2.pdf, …(one per question).

Ensure you have read, signed, and submitted the **Academic Integrity Declaration** AID01.TXT. **Due:** Tuesday Jan 20, 5:00PM **Grace period:** submissions made before 8:00PM on Jan. 20, will be accepted without penalty. Please note that submissions made after 8:00PM **will not be graded** and may only be reviewed for feedback.

Question 1 [6 marks]

There are many different definitions of "little-omega" in the literature (to distinguish them, we will call them $\omega_1, \omega_2, \omega_3$ here). Fix two functions $f(x), g(x)$ from $\mathbb{R}^+$ to $\mathbb{R}^+$; in particular they are never 0. We say that

(i) $f(x) \in \omega_1(g(x))$ if for all $c > 0$ there exists $n_0 > 0$ such that $f(x) > c \cdot g(x)$ for all $x \geq n_0$,

(ii) $f(x) \in \omega_2(g(x))$ if for all $c > 0$ there exists $n_0 > 0$ such that $f(x) \geq c \cdot g(x)$ for all $x \geq n_0$,

(iii) $f(x) \in \omega_3(g(x))$ if the function $\frac{f(x)}{g(x)}$ tends to infinity.

Show that these definitions are equivalent, i.e., $f(x) \in \omega_i(g(x))$ if and only if $f(x) \in \omega_j(g(x))$ for any i, j. Your proof may use the limit-rule (and related statements) only as far as their actual proofs are in the course notes; otherwise you need to re-prove the statement (but you may copy and modify proofs from the course notes).

Recall that the easiest way to prove that a number of statements are equivalent is to prove a circle of implications among them, e.g. (i)⇒(ii)⇒(iii)⇒(i). Picking the circle to prove is up to you, but state clearly what you are proving.

Question 2 [3+6= 9 marks]

(a) Show that the following statement is **true**.

"Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically increasing function. Assume that $f(x) \leq x$ whenever x is a power of 2, i.e., $x = 2^k$ for some integer $k \geq 0$. Then $f(x) \in O(x)$."

(b) Show that the following statement is **false**.

"Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically increasing function. Assume that $f(x) \leq x$ for infinitely many integers, i.e., for any N there exists an integer $x \geq N$ with $f(x) \leq x$. Then $f(x) \in O(x)$."

Reminder: To show that a statement is false, you need to give an example that satisfies all assumptions of the statement, but does not satisfy the conclusion.

Question 3 [2+3+7+2(+1)=14(+1)]

We define the Fibonacci sequence $\{t_n\}$ by $t_0 = 1$, $t_1 = 1$, and for $n \geq 2$,

$$t_n = t_{n-1} + t_{n-2}.$$

(a) Show that $t_n \geq \left(\sqrt{2}\right)^n$ for $n \geq 8$.

(b) Find a constant $k < 1$ such that $t_n \leq 2^{kn}$ for $n \geq 0$. Justify that the inequality holds for your choice of k.

(c) One way to compute t_n uses matrix exponentiation. We can express the linear system

$$\begin{cases} t_1 = t_1 \\ t_2 = t_0 + t_1 \end{cases}$$

in matrix notation:

$$\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \end{bmatrix}.$$

Note that in general,

$$\begin{bmatrix} t_n \\ t_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} t_0 \\ t_1 \end{bmatrix}.$$

Give an algorithm $\mathcal{A}$ to compute t_n that uses $O(\log n)$ 2×2 **matrix multiplications**.

(d) Argue that 4 additions and 8 multiplications (of integers) suffice to compute the product of two 2×2 matrices with integer entries

Note: as a consequence, the runtime of your algorithm $\mathcal{A}$ in (c) is $O(\log n)$).

(e) (Bonus) Another algorithm to compute t_n is

Algorithm 1: $\mathcal{B}$ (int n)

Input: $n \geq 0$

if $n == 0$ **then** return 0
if $n == 1$ **then** return 1
create array of integers $T[0..n]$
$T[0] \leftarrow 0; T[1] \leftarrow 1$
for $i \leftarrow 2$ *to* n **do**
  $T[i] \leftarrow T[i-1] + T[i-2]$
return $T[n]$

Explain why the $O(\log n)$ algorithm $\mathcal{A}$ is likely to be slower than the $\Omega(n)$ algorithm $\mathcal{B}$ when implemented on an actual machine.

Question 4 [3+3+3=9 marks]

Consider the following (rather strange) code-fragment:

Algorithm 2: mystery (int n)

Input: $n \geq 2$

$L \leftarrow \lfloor \log(\log(n)) \rfloor$
print all subsets of $\{1, \ldots, 2^L\}$

For example, for $n = 17$, we have $\log 17 \approx 4.08$ and $\log(4.08) \approx 2.02$, so $\log\log(17) \approx 2.02$ and $L = 2$ (and we print the 16 subsets of $\{1, \ldots, 4\}$). This question is really asking about the run-time of `mystery`, but to avoid having to deal with constants, define $f(n)$ to be the number of subsets that we are printing when calling `mystery` with parameter n.

(a) Show that $f(n) \in O(n)$.

(b) Show that $f(n) \in \Omega(\sqrt{n})$.

(c) Prof. Conn Fused thinks that $f(n) \in \Theta(n^d)$ for some constant d. (By the previous two parts, necessarily $\frac{1}{2} \leq d \leq 1$.) Show that Prof. Fused is wrong, or in other words, for any $\frac{1}{2} \leq d \leq 1$ we have $f(n) \notin \Theta(n^d)$.