1. **[5 marks] Short answer.** You do **not** need to show your work for any part of this question.

   (a) [**1 mark**] Consider a predicate $P(n)$, where $n \in \mathbb{N}$, and suppose that you have proven that $P(2)$ is True, and also that $\forall k \in \mathbb{N}, P(k) \Rightarrow P(2k)$.
   Put an "X" in the box next to **each** statement below that you can conclude to be True.

   Solution

   [X] $P(4)$ [ ] $P(5)$ [ ] $P(6)$ [ ] $P(7)$ [X] $P(8)$ [ ] $P(9)$ [ ] $P(10)$

   (b) [**1 mark**] Consider the natural number n whose decimal representation is $(14)_{10}$.
   Put an "X" in the box next to **each** correct statement below.

   Solution

   [ ] $n = (1011)_2$ [X] $n = (1110)_2$ [X] $n = (112)_3$ [ ] $n = (111)_4$

   (c) [**1 mark**] Put an "X" in the box next to **each** correct statement below about functions $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$ where:

   $$f(n) = 6n + 1 \qquad \text{and} \qquad g(n) = n^2$$

   Solution

   [X] f is eventually dominated by g [ ] g is eventually dominated by f
   [ ] f is dominated by g up to a constant factor [ ] g is dominated by f up to a constant factor
   [ ] $f \in \Omega(g)$ [X] $g \in \Omega(f)$

   (d) [**1 mark**] Let $RT_f(n) : \mathbb{N} \to \mathbb{R}^{\geq 0}$ be the running time function of the following algorithm.

```
def f(n: int) -> None:
    """Precondition: n >= 0."""
    i = 1
    while i < n:
        i = i * 2
```

   Put an "X" in the box next to **each** correct statement below.

   Solution

   [ ] $RT_f(n) \in \mathcal{O}(1)$ [X] $RT_f(n) \in \mathcal{O}(n)$
   [X] $RT_f(n) \in \mathcal{O}(\log_2 n)$ [ ] $RT_f(n) \in \Theta(n)$

   (e) [**1 mark**] Let S be a non-empty finite set of real numbers, and let $m \in \mathbb{R}$.
   Put an "X" in the box next to the expression below that is equivalent to the English statement "m is a lower bound on the minimum value of S"?

Solution

☐ $\forall x \in S, x \leq m$ ☐ $\exists x \in S, x \leq m$ ☒ $\forall x \in S, m \leq x$ ☐ $\exists x \in S, m \leq x$

2. **[5 marks] Induction.**

Prove the following statement using induction.

$$\forall n \in \mathbb{N},\ (n \geq 2) \Rightarrow \left(\prod_{i=2}^{n} \left(1 - \frac{1}{i}\right) = \frac{1}{n} \right)$$

<u>**Solution**</u>

Note: This solution is wordier than expected and provides more intermediate steps than some might find necessary.

Proof. <u>**Base case**</u>: Let $n = 2$. Then

$$\begin{aligned}
\prod_{i=2}^{n} \left(1 - \frac{1}{i}\right) &= \prod_{i=2}^{2} \left(1 - \frac{1}{i}\right) \\
&= \left(1 - \frac{1}{2}\right) \\
&= \frac{1}{2} \\
&= \frac{1}{n},
\end{aligned}$$

as required. The base case is satisfied.

(Or take a 'compute left hand side', 'compute right hand side' approach, and compare.)

<u>**Induction step**</u>: Let $k \in \mathbb{N}$, and assume that $k \geq 2$ and $\prod_{i=2}^{k} \left(1 - \frac{1}{i}\right) = \frac{1}{k}$. We'll prove that $\prod_{i=2}^{k+1} \left(1 - \frac{1}{i}\right) = \frac{1}{(k+1)}$.

We have:

$$\begin{aligned}
\prod_{i=2}^{k+1} \left(1 - \frac{1}{i}\right) &= \left(\prod_{i=2}^{k} \left(1 - \frac{1}{i}\right) \right) \cdot \left(1 - \frac{1}{(k+1)}\right) \\
&= \left(\frac{1}{k}\right) \cdot \left(1 - \frac{1}{(k+1)}\right) \qquad \text{(by the I.H.)} \\
&= \left(\frac{1}{k}\right) \cdot \left(\frac{(k+1) - 1}{(k+1)}\right) \\
&= \left(\frac{1}{k}\right) \cdot \left(\frac{k}{(k+1)}\right) \\
&= \frac{1}{(k+1)},
\end{aligned}$$

as required. □

3. **[5 marks] Asymptotic analysis.**

In this question, refer to the following definition:

$$g \in \mathcal{O}(f): \quad \exists c, n_0 \in \mathbb{R}^+,\ \forall n \in \mathbb{N},\ n \geq n_0 \Rightarrow g(n) \leq c \cdot f(n), \quad \text{where } f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$$

Prove or disprove the following statement, using only the definition of Big-O:

$$\forall f, g : \mathbb{N} \to \mathbb{R}^{\geq 0},\ \Big(\big(\forall n \in \mathbb{N},\ n \geq 165 \Rightarrow g(n) \leq 100\ n\big) \land \big(\forall n \in \mathbb{N},\ n \geq 108 \Rightarrow n \leq 2\ f(n)\big)\Big) \Rightarrow g \in \mathcal{O}(f)$$

Solution

Proof. Let f, g be arbitrary functions from $\mathbb{N} \to \mathbb{R}^{\geq 0}$.

Assume $\forall n \in \mathbb{N}, n \geq 165 \Rightarrow g(n) \leq 100\ n$ and $\forall n \in \mathbb{N}, n \geq 108 \Rightarrow n \leq 2\ f(n)$.

We need to prove $g \in \mathcal{O}(f)$.

That is, we need to prove $\exists c, n_0 \in \mathbb{R}^+,\ \forall n \in \mathbb{N},\ n \geq n_0 \Rightarrow g(n) \leq c \cdot f(n)$

Let $c = 200$ and $n_0 = 165$. (Any $c \geq 200$ or $n_0 \geq 165$ work too.)

Let $n \in \mathbb{N}$ and assume $n \geq n_0$.

Since $n \geq 165$, $g(n) \leq 100\ n$.

Since $n \geq 108$, $n \leq 2\ f(n)$.

Together, we have

$$\begin{aligned} g(n) &\leq 100\ n \\ &\leq 100\ (2\ f(n)) \\ &= 200\ f(n) \\ &= c \cdot f(n), \end{aligned}$$

as required. $\square$

4. **[10 marks] Running time analysis.**

(a) [**4 marks**] Consider the following algorithm.

```
def f(n: int) -> None:
    """Precondition: n >= 2."""
    i = 2
    while i <= n:           # Loop 1
        j = 0
        while j < n:        # Loop 2
            j = j + (n / i)
        i = i + 2
```

Find the **exact total number of iterations of the Loop 2 body, across all iterations of Loop 1** when `f` is run, in terms of its input n, assuming $n \geq 2$. To simplify your calculations, you may ignore floors and ceilings.

Note: make sure to explain your analysis in English, rather than writing only calculations.

Solution

The values of i executing the Loop 1 body are $i = 2, 4, 6, \dots,$ up to and including $n-1$ or n, i.e. $i = 2 \cdot 1, 2 \cdot 2, 2 \cdot 3, \dots, 2 \cdot \lfloor n/2 \rfloor$ (or just $n/2$ ignoring floors and ceilings).

For each i, the values of j executing the Loop 2 body are $j = 0 \cdot (n/i), 1 \cdot (n/i), 2 \cdot (n/i), 3 \cdot (n/i), \dots,$ up to just before $n = i \cdot (n/i)$ (trace with a concrete n and some is for intuition), which is (ignoring floors and ceilings) i iterations.

The total is $\displaystyle\sum_{k=1}^{n/2} 2k = 2\sum_{k=1}^{n/2} k = 2(n/2)(1 + n/2)/2 = n(2+n)/4 = (2n + n^2)/4.$

(b) [**6 marks**] Consider the following algorithm, which takes as input a list of nonnegative integers.

```
def alg(A: list[int]) -> None:
    """Precondition: A is a list of nonnegative integers."""
    for i in range(0, len(A)):             # Loop 1
        if A[i] % 2 == 0:
            for j in range(A[i], len(A)):  # Loop 2
                A[j] = 1
            return
```

NOTE: `range(a, b)` is *empty* when `b <= a`.

Prove matching upper (Big-O) and lower (Omega) bounds on the worst-case running time of `alg`, where the size n of the input is the length of the list. Clearly label which part of your solution is a proof of the upper bound, and which part is a proof of the lower bound.

Solution

Upper bound on worst-case running time.

Let $n \in \mathbb{N}$ and `A` be a list of integers of non-negative integers of length n.

Loop 1 iterates no more than n times.

Its body takes 1 step each time, except at most once if the if condition is true in which case it executes Loop 2 and then ends execution.

Loop 2 executes its body $n - \min(A[i], n)$ times (the minimum is for when there are no iterations due to $A[i] \geq n$). This is at most n times since that minimum is non-negative. The body is 1 step each time. Then there is 1 more step for the return which ends the execution.

So the total number of steps is no more than $n \cdot 1 + n \cdot 1 + 1 = 2n + 1 \in \mathcal{O}(n)$.

Lower bound on worst-case running time.

Let $n \in \mathbb{N}$ and $A = [1, \ldots, 1]$ be the list of length n with non-negative integer 1 for each element.

Then each element is odd so the if condition is always false, so Loop 1 iterates to the end taking 1 step each time, for a total number of steps $n \cdot 1 = n \in \Omega(n)$.