

201A-MATH182-1 HW1

ESHAN UNIYAL

TOTAL POINTS

86 / 90

QUESTION 1

1 4 / 5

- 5 pts No submission
- 0 pts Correct
- 2 pts (2) Incorrect
- ✓ - 1 pts** In (2), k ranges over all integers, including negative integers
- 4 pts (1) and (2) incorrect

QUESTION 2

2 5 / 5

- ✓ - 0 pts** Correct
- 5 pts No submission
- 1 pts Write $\sum_{j=1}^{\infty} j^2 = \sum_{k=1}^{\infty} k^2$
- 3 pts Write $\sum_{k=1}^{\infty} k^2 = \left(\sum_{k=1}^n k^2 \right) + \sum_{k=n+1}^{\infty} k^2$
- 1 pts Errors in calculations
- 1 pts Use geometric sum formula with wrong bounds

QUESTION 3

3 5 / 5

- ✓ - 0 pts** Correct
- 5 pts No submission
- 1 pts Incomplete/Incorrect

QUESTION 4

4 5 / 5

- 5 pts No submission
- ✓ - 0 pts** Correct
- 4 pts No submission for parts (2), (3), (4)
- 2.5 pts No submission for parts (3) and (4)
- 3 pts No submission for parts (3) and (4), analysis in (2) is incomplete

QUESTION 5

5 5 / 5

- ✓ - 0 pts** Correct
- 5 pts No submission
- 1 pts Error in strictness of inequalities
- 1 pts Incomplete
- 2 pts Incorrect
- 3 pts Floor/ceiling are not additive

QUESTION 6

6 5 / 5

- 5 pts No submission
- ✓ - 0 pts** Correct
- 3 pts No condition given
- 3 pts Incorrect
- 2 pts More justification needed

QUESTION 7

7 5 / 5

- 5 pts No submission
- ✓ - 0 pts** Correct
- 3 pts Incorrect formula, not in closed form
- 2 pts Correct formula, not in closed form
- 1 pts Error
- 2 pts Incorrect closed formula

QUESTION 8

8 5 / 5

- 5 pts No submission
- ✓ - 0 pts** Correct
- 1 pts Error
- 3 pts Incomplete
- 2 pts Incorrect argument
- 2 pts No final formula given
- 2 pts more justification needed

QUESTION 9

9 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Neglect to verify that $\$0 \leq z(x) \text{ mod } y < zy$.

QUESTION 10

10 3 / 5

- 0 pts Correct

- 5 pts No submission

- 2 pts Assert $\$a \text{ mod } r = a \text{ mod } rs$.

✓ - 2 pts Assume that $\$x \mapsto x \text{ mod } r$ is additive.

- 1 pts Missing step

1 It is not, e.g. $\$1+1 \text{ mod } 2 = 0$.

2 "Closed under addition" is a property of a set, that if you add two numbers in the set you get a number in the set. The term you want it "additive", or "distributes over addition", or "commutes with addition".

QUESTION 11

11 5 / 5

✓ - 0 pts Correct.

- 5 pts No submission

- 2 pts Incorrect

QUESTION 12

12 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Program given without answer.

- 0 pts Wrong answer because of integer overflow

QUESTION 13

13 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Fencepost error; added an extra Fibonacci number.

- 1 pts Incorrectly initialize sum.

QUESTION 14

14 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Give program without answer.

- 2 pts Serious error

QUESTION 15

15 5 / 5

- 5 pts No submission

- 2 pts Incorrect answer, needed to find P_{100000} (one-hundred thousandth prime), not P_{10000} (ten-thousandth prime)

✓ - 0 pts Correct

- 1 pts Error

- 2 pts Big error

QUESTION 16

16 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Calculate number of distinct remainders instead of cycle length.

- 2 pts Incorrect

3 You don't really need to check whether division has terminated. A zero remainder produces a cycle of length 1.

4 Nice. I like this optimization.

QUESTION 17

17 5 / 5

- 5 pts No submission

✓ - 0 pts Correct

- 1 pts Incomplete

- 2 pts Incomplete/Incorrect

- 2 pts Using asymptotic notation incorrectly

QUESTION 18

18 4 / 5

- **5 pts** No submission

- **0 pts** Correct

✓ - **1 pts** Need to use induction, cannot include ... in a
trail of equalities.

- **1 pts** Circular reasoning

- **1 pts** Error in argument

- **2 pts** Incorrect use of log identity

- **3 pts** No argument given

- **2 pts** More rigor needed in argument

MATH182 HOMEWORK #1
DUE June 18, 2020

Note: while you are encouraged to work together on these problems with your classmates, your final work should be written in your own words and not copied verbatim from somewhere else. You need to do at least seven (7) of these problems. All problems will be graded, although the score for the homework will be capped at $N := (\text{point value of one problem}) \times 7$ and the homework will be counted out of N total points. Thus doing more problems can only help your homework score. For the programming exercise you should submit the final answer (a number) *and* your program source code.

Exercise 1. Write out the following two sums in full:

$$(1) \sum_{0 \leq k \leq 5} a_k$$

$$(2) \sum_{0 \leq k^2 \leq 5} a_{k^2}$$

$$(1) \sum_{0 \leq k \leq 5} a_k = a_0 + a_1 + a_2 + a_3 + a_4 + a_5$$

$$(2) \sum_{0 \leq k^2 \leq 5} a_{k^2} = a_0 + a_1 + a_4$$

Exercise 2. Evaluate the following summation:

$$\sum_{k=1}^n k2^k.$$

Hint: rewrite as a double sum.

Expressing k as $\sum_{i=1}^k 1$, we have

$$\sum_{k=1}^n k2^k = \sum_{k=1}^n \left(\sum_{i=1}^k 1 \cdot 2^k \right) = \sum_{k=1}^n \sum_{i=1}^k 2^k$$

In the above double sum, we have $1 \leq i \leq k \leq n$. Changing the order of the double sum, we get:

$$\sum_{k=1}^n k2^k = \sum_{k=1}^n \sum_{i=1}^k 2^k = \sum_{i=1}^n \sum_{k=i}^n 2^k$$

We can now solve the double sum using the formula for a geometric series:

$$\begin{aligned} \therefore \sum_{k=1}^n k2^k &= \sum_{i=1}^n \sum_{k=i}^n 2^k = \sum_{i=1}^n \left(\sum_{k=0}^n 2^k - \sum_{k=0}^{i-1} 2^k \right) \\ &= \sum_{i=1}^n \left(\frac{1-2^{n+1}}{1-2} - \frac{1-2^i}{1-2} \right) = \sum_{i=1}^n (2^{n+1} - 2^i) = \sum_{i=1}^n 2^{n+1} - \sum_{i=1}^n 2^i \\ &= 2^{n+1} \sum_{i=1}^n 1 - \left(\sum_{i=0}^n 2^i - 2^0 \right) = 2^{n+1} \cdot n - \left(\frac{1-2^{n+1}}{1-2} - 1 \right) \\ &= n2^{n+1} - (2^{n+1} - 2) = (n-1)2^{n+1} + 2 \end{aligned}$$

$$\therefore \sum_{k=1}^n k2^k = (n-1)2^{n+1} + 2$$

1 4 / 5

- 5 pts No submission
 - 0 pts Correct
 - 2 pts (2) Incorrect
- ✓ - 1 pts In (2), k ranges over all integers, including negative integers
- 4 pts (1) and (2) incorrect

MATH182 HOMEWORK #1
DUE June 18, 2020

Note: while you are encouraged to work together on these problems with your classmates, your final work should be written in your own words and not copied verbatim from somewhere else. You need to do at least seven (7) of these problems. All problems will be graded, although the score for the homework will be capped at $N := (\text{point value of one problem}) \times 7$ and the homework will be counted out of N total points. Thus doing more problems can only help your homework score. For the programming exercise you should submit the final answer (a number) *and* your program source code.

Exercise 1. Write out the following two sums in full:

$$(1) \sum_{0 \leq k \leq 5} a_k$$

$$(2) \sum_{0 \leq k^2 \leq 5} a_{k^2}$$

$$(1) \sum_{0 \leq k \leq 5} a_k = a_0 + a_1 + a_2 + a_3 + a_4 + a_5$$

$$(2) \sum_{0 \leq k^2 \leq 5} a_{k^2} = a_0 + a_1 + a_4$$

Exercise 2. Evaluate the following summation:

$$\sum_{k=1}^n k2^k.$$

Hint: rewrite as a double sum.

Expressing k as $\sum_{i=1}^k 1$, we have

$$\sum_{k=1}^n k2^k = \sum_{k=1}^n \left(\sum_{i=1}^k 1 \cdot 2^k \right) = \sum_{k=1}^n \sum_{i=1}^k 2^k$$

In the above double sum, we have $1 \leq i \leq k \leq n$. Changing the order of the double sum, we get:

$$\sum_{k=1}^n k2^k = \sum_{k=1}^n \sum_{i=1}^k 2^k = \sum_{i=1}^n \sum_{k=i}^n 2^k$$

We can now solve the double sum using the formula for a geometric series:

$$\begin{aligned} \therefore \sum_{k=1}^n k2^k &= \sum_{i=1}^n \sum_{k=i}^n 2^k = \sum_{i=1}^n \left(\sum_{k=0}^n 2^k - \sum_{k=0}^{i-1} 2^k \right) \\ &= \sum_{i=1}^n \left(\frac{1-2^{n+1}}{1-2} - \frac{1-2^i}{1-2} \right) = \sum_{i=1}^n (2^{n+1} - 2^i) = \sum_{i=1}^n 2^{n+1} - \sum_{i=1}^n 2^i \\ &= 2^{n+1} \sum_{i=1}^n 1 - \left(\sum_{i=0}^n 2^i - 2^0 \right) = 2^{n+1} \cdot n - \left(\frac{1-2^{n+1}}{1-2} - 1 \right) \\ &= n2^{n+1} - (2^{n+1} - 2) = (n-1)2^{n+1} + 2 \end{aligned}$$

$$\therefore \sum_{k=1}^n k2^k = (n-1)2^{n+1} + 2$$

2 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Write $\sum_{j=1}^n 2^k = \sum_{j=1}^n 2^j$.

- 3 pts Write $\sum_{k=1}^n k 2^k = \left(\sum_{k=1}^n k \right) \left(\sum_{k=1}^n 2^k \right)$.

- 1 pts Errors in calculations

- 1 pts Use geometric sum formula with wrong bounds

Exercise 3. Suppose $x \neq 1$. Prove that

$$\sum_{j=0}^n jx^j = \frac{nx^{n+1} - (n+1)x^{n+1} + x}{(x-1)^2}.$$

Challenge: do this without using mathematical induction.

We first extract the first term of the series and express j as $\sum_{i=1}^j 1$ to get:

$$\sum_{j=0}^n jx^j = 0 \cdot x^0 + \sum_{j=1}^n jx^j = \sum_{j=1}^n \left(\sum_{i=1}^j 1 \cdot x^j \right) = \sum_{j=1}^n \sum_{i=1}^j x^j$$

In the above double sum, we have $1 \leq i \leq j \leq n$. Changing the order of the double sum, we get:

$$\sum_{j=1}^n jx^k = \sum_{j=1}^n \sum_{i=0}^n x^j = \sum_{i=1}^n \sum_{j=i}^n x^j$$

We can now compute the summation as in Exercise 2:

$$\begin{aligned} \sum_{j=1}^n jx^k &= \sum_{i=1}^n \sum_{j=i}^n x^j = \sum_{i=1}^n \left(\sum_{j=0}^n x^j - \sum_{j=0}^{i-1} x^j \right) \\ &= \sum_{i=1}^n \left(\frac{1-x^{n+1}}{1-x} - \frac{1-x^i}{1-x} \right) = \frac{1}{1-x} \sum_{i=1}^n (x^i - x^{n+1}) \\ &= \frac{1}{1-x} \left(\sum_{i=1}^n x^i - \sum_{i=1}^n x^{n+1} \right) = \frac{1}{1-x} \left(\sum_{i=0}^n x^i - x^0 - x^{n+1} \sum_{i=1}^n 1 \right) \\ &= \frac{1}{1-x} \left(\frac{1-x^{n+1}}{1-x} - 1 - nx^{n+1} \right) = \frac{1-x^{n+1} - (1-x) - (1-x)nx^{n+1}}{(1-x)^2} \\ &= \frac{-x^{n+1} + x - nx^{n+1} + nx^{n+2}}{(1-x)^2} = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2} \\ \therefore \quad \sum_{j=1}^n jx^k &= \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2} \quad \blacksquare \end{aligned}$$

Exercise 4 (Recommended! Horner's rule). The following code fragment implements Horner's rule for evaluating a polynomial

$$\begin{aligned} P(x) &= \sum_{k=0}^n a_k x^k \\ &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n))), \end{aligned}$$

given coefficients a_0, a_1, \dots, a_n and a value for x :

```

1  y = 0
2  for i = n downto 0
3      y = a_i + x · y

```

- (1) In terms of Θ -notation, what is the running time of this code fragment for Horner's rule?
- (2) Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare to Horner's rule?

3 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Incomplete/Incorrect

Exercise 3. Suppose $x \neq 1$. Prove that

$$\sum_{j=0}^n jx^j = \frac{nx^{n+1} - (n+1)x^{n+1} + x}{(x-1)^2}.$$

Challenge: do this without using mathematical induction.

We first extract the first term of the series and express j as $\sum_{i=1}^j 1$ to get:

$$\sum_{j=0}^n jx^j = 0 \cdot x^0 + \sum_{j=1}^n jx^j = \sum_{j=1}^n \left(\sum_{i=1}^j 1 \cdot x^j \right) = \sum_{j=1}^n \sum_{i=1}^j x^j$$

In the above double sum, we have $1 \leq i \leq j \leq n$. Changing the order of the double sum, we get:

$$\sum_{j=1}^n jx^k = \sum_{j=1}^n \sum_{i=0}^n x^j = \sum_{i=1}^n \sum_{j=i}^n x^j$$

We can now compute the summation as in Exercise 2:

$$\begin{aligned} \sum_{j=1}^n jx^k &= \sum_{i=1}^n \sum_{j=i}^n x^j = \sum_{i=1}^n \left(\sum_{j=0}^n x^j - \sum_{j=0}^{i-1} x^j \right) \\ &= \sum_{i=1}^n \left(\frac{1-x^{n+1}}{1-x} - \frac{1-x^i}{1-x} \right) = \frac{1}{1-x} \sum_{i=1}^n (x^i - x^{n+1}) \\ &= \frac{1}{1-x} \left(\sum_{i=1}^n x^i - \sum_{i=1}^n x^{n+1} \right) = \frac{1}{1-x} \left(\sum_{i=0}^n x^i - x^0 - x^{n+1} \sum_{i=1}^n 1 \right) \\ &= \frac{1}{1-x} \left(\frac{1-x^{n+1}}{1-x} - 1 - nx^{n+1} \right) = \frac{1-x^{n+1} - (1-x) - (1-x)nx^{n+1}}{(1-x)^2} \\ &= \frac{-x^{n+1} + x - nx^{n+1} + nx^{n+2}}{(1-x)^2} = \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2} \\ \therefore \quad \sum_{j=1}^n jx^k &= \frac{nx^{n+2} - (n+1)x^{n+1} + x}{(1-x)^2} \quad \blacksquare \end{aligned}$$

Exercise 4 (Recommended! Horner's rule). The following code fragment implements Horner's rule for evaluating a polynomial

$$\begin{aligned} P(x) &= \sum_{k=0}^n a_k x^k \\ &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x a_n))), \end{aligned}$$

given coefficients a_0, a_1, \dots, a_n and a value for x :

```

1  y = 0
2  for i = n downto 0
3      y = a_i + x · y

```

- (1) In terms of Θ -notation, what is the running time of this code fragment for Horner's rule?
- (2) Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare to Horner's rule?

(3) Consider the following loop invariant:

At the start of each iteration of the **for** loop of lines 2-3

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k.$$

Interpret a summation with no terms as equaling 0. Following the structure of the loop invariant proof presented in this chapter, use this loop invariant to show that, at termination, $y = \sum_{k=0}^n a_k x^k$.

(4) Conclude by arguing that the given code fragment correctly evaluates a polynomial characterized by the coefficients a_0, a_1, \dots, a_n .

- (1) Since the this code fragment implements a loop that runs $n + 1$ times and takes a constant amount of time each iteration, its running time is $\Theta(n)$.
- (2) Below is the pseudocode for a naive polynomial-evaluation algorithm:

```

1    $y = 0$ 
2   for  $i = 0$  to  $n$ 
3        $y = y + a_i \cdot x^i$ 
```

Since this code fragment too implements a loop that runs $n + 1$ times and seems to take a constant amount of time each iteration, its running time should also be $\Theta(n)$. However, this analysis assumes that evaluating x^i is a constant-time operation — in fact, however, evaluating x^i is $\Theta(i)$ with a simple loop (or $\Theta(i \log i)$ with some optimisations). The actual running time of the naive algorithm is therefore $\Theta(n^2)$ or $\Theta(n \log n)$ (depending on the implementation of the power function). In either case, Horner's rule is more efficient.

(3) **Initialisation:** Before the first iteration, we have $y = 0$ and $i = n$. We compute the right hand side of the loop invariant equation:

$$\sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k = \sum_{k=0}^{n-(n+1)} a_{k+n+1} x^k = \sum_{k=0}^{-1} a_{k+n+1} x^k = 0 = y$$

Initialisation is therefore correct.

Maintenance: Assume the loop invariant is true before some iteration with $i = j$, $0 < j \leq n$. We have from, the loop invariant, $y = \sum_{k=0}^{n-(j+1)} a_{k+j+1} x^k$. After running line 3 with $i = j$, we now get:

$$\begin{aligned}
y &:= a_j + x \cdot y = a_j + x \sum_{k=0}^{n-(j+1)} a_{k+j+1} x^k \\
&= a_j + \sum_{k=0}^{n-(j+1)} a_{k+j+1} x^{k+1} = a_j + \sum_{k=1}^{n-(j+1)+1} a_{k+j} x^k \\
&= a_{0+j} x^0 + \sum_{k=1}^{n-j} a_{k+j} x^k = \sum_{k=0}^{n-((j-1)+1)} a_{k+(j-1)+1} x^k \\
\therefore y &= \sum_{k=0}^{n-((j-1)+1)} a_{k+(j-1)+1} x^k
\end{aligned}$$

In other words, the loop invariant is true before the body of the next iteration, in which $i = j - 1$, as desired.

Termination: In the final iteration, we have $i = -1$. From the loop invariant, we have

$$y = \sum_{k=0}^{n-(-1+1)} a_{k-1+1}x^k = \sum_{k=0}^n a_kx^k \implies y = \sum_{k=0}^n a_kx^k$$

- (4) The body of the loop isn't run in the final iteration, since $i = -1 \notin [0, n]$. The program then returns the last state of y , $\sum_{k=0}^n a_kx^k$, which is exactly the polynomial $a_0 + a_1x + \dots + a_nx^n$. The code fragment therefore correct.

Exercise 5. Suppose $m, n \in \mathbb{Z}$ are such that $m > 0$. Prove that

$$\left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil$$

This gives us another reflection principle between floors and ceilings when the argument is a rational number.

We prove the statement directly. Let $k := \left\lceil \frac{n}{m} \right\rceil$. By definition of the ceiling function, we have

$$k - 1 < \frac{n}{m} \leq k \implies (k-1)m < n \leq km \quad (\because m > 0)$$

Since $k, m, n \in \mathbb{Z}$, each part of the inequality is an integer. Manipulating the inequality, we have:

$$(k-1)m \leq n - 1 < km \quad (\because (k-1)m < n, n \leq km)$$

Dividing all sides by m and adding 1, we get:

$$k \leq \frac{n-1}{m} + 1 < k + 1 \implies k \leq \frac{n+m-1}{m} < k + 1$$

Since the second half of this final inequality is strict, by the definition of the floor function, it is sufficient to conclude:

$$\left\lceil \frac{n+m-1}{m} \right\rceil = k \implies \left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil \blacksquare$$

Exercise 6. Find a necessary and sufficient condition on the real number $b > 1$ such that

$$\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor$$

holds for all real numbers $x \geq 1$.

We show that $b \in \mathbb{Z}$ (in addition to the given assumption $b > 1$) is a necessary and sufficient condition for the above equality for all $x \geq 1$. Let $x := b \geq 1$ (by assumption on b). From the equality, we then have:

$$\lfloor \log_b \lfloor x \rfloor \rfloor = \lfloor \log_b x \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor = \lfloor \log_b b \rfloor = \lfloor 1 \rfloor = 1$$

Assume towards contradiction that $b \notin \mathbb{Z}$. By definition of the floor function, we therefore have $\lfloor b \rfloor < b$. Since the logarithm function is continuous and increasing for base $b > 1$, taking \log_b on both sides maintains the inequality and we get $\log_b \lfloor b \rfloor < \log_b b = 1$.

Since the right hand side of this inequality is integral, we have $\lfloor \log_b b \rfloor = \log_b b$. From the definition of the floor function, we extend the inequality to get

$$\lfloor \log_b \lfloor b \rfloor \rfloor \leq \log_b \lfloor b \rfloor < \log_b b = \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor < \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor \neq \lfloor \log_b b \rfloor$$

4 5 / 5

- 5 pts No submission
- ✓ - 0 pts Correct
- 4 pts No submission for parts (2), (3), (4)
- 2.5 pts No submission for parts (3) and (4)
- 3 pts No submission for parts (3) and (4), analysis in (2) is incomplete

In other words, the loop invariant is true before the body of the next iteration, in which $i = j - 1$, as desired.

Termination: In the final iteration, we have $i = -1$. From the loop invariant, we have

$$y = \sum_{k=0}^{n-(-1+1)} a_{k-1+1}x^k = \sum_{k=0}^n a_kx^k \implies y = \sum_{k=0}^n a_kx^k$$

- (4) The body of the loop isn't run in the final iteration, since $i = -1 \notin [0, n]$. The program then returns the last state of y , $\sum_{k=0}^n a_kx^k$, which is exactly the polynomial $a_0 + a_1x + \dots + a_nx^n$. The code fragment therefore correct.

Exercise 5. Suppose $m, n \in \mathbb{Z}$ are such that $m > 0$. Prove that

$$\left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil$$

This gives us another reflection principle between floors and ceilings when the argument is a rational number.

We prove the statement directly. Let $k := \left\lceil \frac{n}{m} \right\rceil$. By definition of the ceiling function, we have

$$k - 1 < \frac{n}{m} \leq k \implies (k-1)m < n \leq km \quad (\because m > 0)$$

Since $k, m, n \in \mathbb{Z}$, each part of the inequality is an integer. Manipulating the inequality, we have:

$$(k-1)m \leq n - 1 < km \quad (\because (k-1)m < n, n \leq km)$$

Dividing all sides by m and adding 1, we get:

$$k \leq \frac{n-1}{m} + 1 < k + 1 \implies k \leq \frac{n+m-1}{m} < k + 1$$

Since the second half of this final inequality is strict, by the definition of the floor function, it is sufficient to conclude:

$$\left\lceil \frac{n+m-1}{m} \right\rceil = k \implies \left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil \blacksquare$$

Exercise 6. Find a necessary and sufficient condition on the real number $b > 1$ such that

$$\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor$$

holds for all real numbers $x \geq 1$.

We show that $b \in \mathbb{Z}$ (in addition to the given assumption $b > 1$) is a necessary and sufficient condition for the above equality for all $x \geq 1$. Let $x := b \geq 1$ (by assumption on b). From the equality, we then have:

$$\lfloor \log_b \lfloor x \rfloor \rfloor = \lfloor \log_b x \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor = \lfloor \log_b b \rfloor = \lfloor 1 \rfloor = 1$$

Assume towards contradiction that $b \notin \mathbb{Z}$. By definition of the floor function, we therefore have $\lfloor b \rfloor < b$. Since the logarithm function is continuous and increasing for base $b > 1$, taking \log_b on both sides maintains the inequality and we get $\log_b \lfloor b \rfloor < \log_b b = 1$.

Since the right hand side of this inequality is integral, we have $\lfloor \log_b b \rfloor = \log_b b$. From the definition of the floor function, we extend the inequality to get

$$\lfloor \log_b \lfloor b \rfloor \rfloor \leq \log_b \lfloor b \rfloor < \log_b b = \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor < \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor \neq \lfloor \log_b b \rfloor$$

5 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Error in strictness of inequalities

- 1 pts Incomplete

- 2 pts Incorrect

- 3 pts Floor/ceiling are not additive

In other words, the loop invariant is true before the body of the next iteration, in which $i = j - 1$, as desired.

Termination: In the final iteration, we have $i = -1$. From the loop invariant, we have

$$y = \sum_{k=0}^{n-(-1+1)} a_{k-1+1}x^k = \sum_{k=0}^n a_kx^k \implies y = \sum_{k=0}^n a_kx^k$$

- (4) The body of the loop isn't run in the final iteration, since $i = -1 \notin [0, n]$. The program then returns the last state of y , $\sum_{k=0}^n a_kx^k$, which is exactly the polynomial $a_0 + a_1x + \dots + a_nx^n$. The code fragment therefore correct.

Exercise 5. Suppose $m, n \in \mathbb{Z}$ are such that $m > 0$. Prove that

$$\left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil$$

This gives us another reflection principle between floors and ceilings when the argument is a rational number.

We prove the statement directly. Let $k := \left\lceil \frac{n}{m} \right\rceil$. By definition of the ceiling function, we have

$$k - 1 < \frac{n}{m} \leq k \implies (k-1)m < n \leq km \quad (\because m > 0)$$

Since $k, m, n \in \mathbb{Z}$, each part of the inequality is an integer. Manipulating the inequality, we have:

$$(k-1)m \leq n - 1 < km \quad (\because (k-1)m < n, n \leq km)$$

Dividing all sides by m and adding 1, we get:

$$k \leq \frac{n-1}{m} + 1 < k + 1 \implies k \leq \frac{n+m-1}{m} < k + 1$$

Since the second half of this final inequality is strict, by the definition of the floor function, it is sufficient to conclude:

$$\left\lceil \frac{n+m-1}{m} \right\rceil = k \implies \left\lceil \frac{n}{m} \right\rceil = \left\lceil \frac{n+m-1}{m} \right\rceil \blacksquare$$

Exercise 6. Find a necessary and sufficient condition on the real number $b > 1$ such that

$$\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor$$

holds for all real numbers $x \geq 1$.

We show that $b \in \mathbb{Z}$ (in addition to the given assumption $b > 1$) is a necessary and sufficient condition for the above equality for all $x \geq 1$. Let $x := b \geq 1$ (by assumption on b). From the equality, we then have:

$$\lfloor \log_b \lfloor x \rfloor \rfloor = \lfloor \log_b x \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor = \lfloor \log_b b \rfloor = \lfloor 1 \rfloor = 1$$

Assume towards contradiction that $b \notin \mathbb{Z}$. By definition of the floor function, we therefore have $\lfloor b \rfloor < b$. Since the logarithm function is continuous and increasing for base $b > 1$, taking \log_b on both sides maintains the inequality and we get $\log_b \lfloor b \rfloor < \log_b b = 1$.

Since the right hand side of this inequality is integral, we have $\lfloor \log_b b \rfloor = \log_b b$. From the definition of the floor function, we extend the inequality to get

$$\lfloor \log_b \lfloor b \rfloor \rfloor \leq \log_b \lfloor b \rfloor < \log_b b = \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor < \lfloor \log_b b \rfloor \implies \lfloor \log_b \lfloor b \rfloor \rfloor \neq \lfloor \log_b b \rfloor$$

This contradicts our assumption that the equality $\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor$ holds for all $x \geq 1$, since it does not hold for $x := b > 1$. We conclude b is necessarily an integer; in particular, $b \in \mathbb{Z}$, $b > 1$.

We now show that b is an integer is also a sufficient condition for the above equality. Let $x \in \mathbb{R}$, $x \geq 1$ be arbitrary, and assume $b \in \mathbb{Z}$, $b > 1$. Define $k := \lfloor \log_b x \rfloor$. From the definition of the floor function, we have the inequality $k \leq \log_b x < k + 1$. Since $b > 1$ (by assumption), the power function b^x is increasing and the inequality is preserved under exponentiation. Raising b to each part of the inequality, we get:

$$b^k \leq b^{\log_b x} < b^{k+1} \implies b^k \leq x < b^{k+1}$$

Since $b, k \in \mathbb{Z}$, we know $b^k, b^{k+1} \in \mathbb{Z}$. Applying the definition of the floor function and once more taking $\log_b b$ on all sides, we get:

$$b^k \leq x < b^{k+1} \implies b^k \leq \lfloor x \rfloor < b^{k+1} \implies k \leq \log_b \lfloor x \rfloor < k + 1 \implies \lfloor \log_b \lfloor x \rfloor \rfloor = k$$

which, by definition of k gives our desired result, $\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor \forall x \in \mathbb{R}, x > 1$. ■

Exercise 7. Suppose $0 < \alpha < \beta$ and $0 < x$ are real numbers. Find a closed formula for the sum of all integer multiples of x in the closed interval $[\alpha, \beta]$.

Define S to be the desired sum. By definition of the mod operator, the greatest multiple of x less than or equal to α is given by $\alpha - \alpha \bmod x$. The first multiple of x in the interval $[\alpha, \beta]$ is therefore given by $m_1 := \alpha - \alpha \bmod x + x$. Similarly, the last multiple of x in the interval is given by $m_n := \beta - \beta \bmod x$. Let m_i be the i^{th} multiple of x in the interval, given by $m_i = m_1 + (i-1)x$. Since we already know the last and first multiple, m_n and m_1 , we can compute n as follows:

$$n = \frac{m_n - m_1}{x} = \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x + x)}{x} = \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x}{x}$$

We can now compute a closed formula for the sum of this series from 1 to n :

$$\begin{aligned} S &= \sum_{i=1}^n m_i = \sum_{i=1}^n (m_1 + (i-1)x) = \sum_{i=1}^n m_1 + \sum_{i=1}^n (i-1)x \\ &= m_1 \sum_{i=1}^n 1 + x \sum_{i=1}^n (i-1) = m_1 \cdot n + x \sum_{i=0}^{n-1} i = m_1 \cdot n + x \frac{(n-1)n}{2} = \frac{n}{2}(2m_1 + nx - x) \\ &= \frac{n}{2} \left(2(\alpha - \alpha \bmod x + x) + \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x}{x} \cdot x - x \right) \\ &= \frac{n}{2}(2\alpha - 2 \cdot \alpha \bmod x + 2x + \beta - \alpha + \alpha \bmod x - \beta \bmod x - x - x) \\ &= \frac{n}{2}((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x)) \\ &= \frac{((\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x)((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2x} \\ &= \frac{\beta\alpha + \beta^2 - \beta \cdot \alpha \bmod x - \beta \cdot \beta \bmod x}{2x} + \frac{-\alpha^2 - \alpha\beta + \alpha \cdot \alpha \bmod x + \alpha \cdot \beta \bmod x}{2x} \\ &\quad + \frac{-\alpha \cdot \beta \bmod x - \beta \cdot \beta \bmod x + \alpha \bmod x \cdot \beta \bmod x + (\beta \bmod x)^2}{2x} \\ &\quad + \frac{\alpha \cdot \alpha \bmod x + \beta \cdot \alpha \bmod x - (\alpha \bmod x)^2 - \alpha \bmod x \cdot \beta \bmod x}{2x} \\ &\quad - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))x}{2x} \end{aligned}$$

6 5 / 5

- 5 pts No submission
- ✓ - 0 pts Correct
- 3 pts No condition given
- 3 pts Incorrect
- 2 pts More justification needed

This contradicts our assumption that the equality $\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor$ holds for all $x \geq 1$, since it does not hold for $x := b > 1$. We conclude b is necessarily an integer; in particular, $b \in \mathbb{Z}$, $b > 1$.

We now show that b is an integer is also a sufficient condition for the above equality. Let $x \in \mathbb{R}$, $x \geq 1$ be arbitrary, and assume $b \in \mathbb{Z}$, $b > 1$. Define $k := \lfloor \log_b x \rfloor$. From the definition of the floor function, we have the inequality $k \leq \log_b x < k + 1$. Since $b > 1$ (by assumption), the power function b^x is increasing and the inequality is preserved under exponentiation. Raising b to each part of the inequality, we get:

$$b^k \leq b^{\log_b x} < b^{k+1} \implies b^k \leq x < b^{k+1}$$

Since $b, k \in \mathbb{Z}$, we know $b^k, b^{k+1} \in \mathbb{Z}$. Applying the definition of the floor function and once more taking $\log_b b$ on all sides, we get:

$$b^k \leq x < b^{k+1} \implies b^k \leq \lfloor x \rfloor < b^{k+1} \implies k \leq \log_b \lfloor x \rfloor < k + 1 \implies \lfloor \log_b \lfloor x \rfloor \rfloor = k$$

which, by definition of k gives our desired result, $\lfloor \log_b x \rfloor = \lfloor \log_b \lfloor x \rfloor \rfloor \forall x \in \mathbb{R}, x > 1$. ■

Exercise 7. Suppose $0 < \alpha < \beta$ and $0 < x$ are real numbers. Find a closed formula for the sum of all integer multiples of x in the closed interval $[\alpha, \beta]$.

Define S to be the desired sum. By definition of the mod operator, the greatest multiple of x less than or equal to α is given by $\alpha - \alpha \bmod x$. The first multiple of x in the interval $[\alpha, \beta]$ is therefore given by $m_1 := \alpha - \alpha \bmod x + x$. Similarly, the last multiple of x in the interval is given by $m_n := \beta - \beta \bmod x$. Let m_i be the i^{th} multiple of x in the interval, given by $m_i = m_1 + (i-1)x$. Since we already know the last and first multiple, m_n and m_1 , we can compute n as follows:

$$n = \frac{m_n - m_1}{x} = \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x + x)}{x} = \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x}{x}$$

We can now compute a closed formula for the sum of this series from 1 to n :

$$\begin{aligned} S &= \sum_{i=1}^n m_i = \sum_{i=1}^n (m_1 + (i-1)x) = \sum_{i=1}^n m_1 + \sum_{i=1}^n (i-1)x \\ &= m_1 \sum_{i=1}^n 1 + x \sum_{i=1}^n (i-1) = m_1 \cdot n + x \sum_{i=0}^{n-1} i = m_1 \cdot n + x \frac{(n-1)n}{2} = \frac{n}{2}(2m_1 + nx - x) \\ &= \frac{n}{2} \left(2(\alpha - \alpha \bmod x + x) + \frac{(\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x}{x} \cdot x - x \right) \\ &= \frac{n}{2}(2\alpha - 2 \cdot \alpha \bmod x + 2x + \beta - \alpha + \alpha \bmod x - \beta \bmod x - x - x) \\ &= \frac{n}{2}((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x)) \\ &= \frac{((\beta - \beta \bmod x) - (\alpha - \alpha \bmod x) - x)((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2x} \\ &= \frac{\beta\alpha + \beta^2 - \beta \cdot \alpha \bmod x - \beta \cdot \beta \bmod x}{2x} + \frac{-\alpha^2 - \alpha\beta + \alpha \cdot \alpha \bmod x + \alpha \cdot \beta \bmod x}{2x} \\ &\quad + \frac{-\alpha \cdot \beta \bmod x - \beta \cdot \beta \bmod x + \alpha \bmod x \cdot \beta \bmod x + (\beta \bmod x)^2}{2x} \\ &\quad + \frac{\alpha \cdot \alpha \bmod x + \beta \cdot \alpha \bmod x - (\alpha \bmod x)^2 - \alpha \bmod x \cdot \beta \bmod x}{2x} \\ &\quad - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))x}{2x} \end{aligned}$$

Simplifying further, we get the closed form expression for the sum of all integer multiples of x in $[\alpha, \beta]$:

$$S = \frac{\beta^2 - \alpha^2 + 2(\alpha \cdot \alpha \bmod x - \beta \cdot \beta \bmod x) + (\beta \bmod x)^2 - (\alpha \bmod x)^2}{2x} - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2}$$

Exercise 8. How many of the numbers 2^m , for $0 \leq m \leq M$ (where $m, M \in \mathbb{N}$), have leading digit 1 when written in decimal notation? Your answer should be a closed formula.

We first note that the number of digits in 2^m in decimal notation is given by $\lfloor \log_{10} 2^m \rfloor$. It is trivial to show that for any $M \in \mathbb{N}$, $0 \leq m := 0 \leq M$ and $2^0 = 1$ has leading digit 1. It is clear that each time there is a strict increase (necessarily by 1) in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$, the number of digits in 2^m increases by 1 and we have 2^m with leading digit 1 and vice-versa.

The number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ is therefore given by the number of unique terms in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$. Since the first term of this sequence is $\lfloor \log_{10} 2^1 \rfloor$ and the last term is $\lfloor \log_{10} 2^M \rfloor$, and each increase in the sequence is necessarily 1 and corresponds to a power of 2 that has leading digit 1, the number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ with leading digit 1 is given by

$$\lfloor \log_{10} 2^M \rfloor - \lfloor \log_{10} 2^1 \rfloor = \lfloor \log_{10} 2^M \rfloor - 0 = \lfloor \log_{10} 2^M \rfloor$$

Including the case where $m = 0$, we therefore have:

The number of terms in the sequence $(2^m)_{0 \leq m \leq M}$ that have leading digit 1 is given by $1 + \lfloor \log_{10} 2^M \rfloor$.

Exercise 9. Suppose $x, y, z \in \mathbb{Z}$ are such that $y, z \geq 1$. Prove that $z(x \bmod y) = (zx) \bmod(zy)$.

Define $k_1 := x \bmod y \in [0, y)$, $k_2 := (zx) \bmod(zy) \in [0, zy)$. By the definition of the mod operator, we have:

$$x = \left\lfloor \frac{x}{y} \right\rfloor y + k_1 ; \quad zx = \left\lfloor \frac{zx}{zy} \right\rfloor zy + k_2 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2$$

Plugging the expression for x from the first equation into the second, we get:

$$z \left(\left\lfloor \frac{x}{y} \right\rfloor y + k_1 \right) = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies \left\lfloor \frac{x}{y} \right\rfloor zy + zk_1 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies zk_1 = k_2$$

By definition of k_1 and k_2 , we now have $z(x \bmod y) = (zx) \bmod(zy)$. ■

Exercise 10. Suppose $a, b, r, s \in \mathbb{Z}$ are such that $r, s \geq 1$. Prove that if $a \bmod rs = b \bmod rs$, then $a \bmod r = b \bmod r$ and $a \bmod s = b \bmod s$.

Let $k := a \bmod(rs) = b \bmod(rs) \in [0, rs)$. By the definition of the module operator, we have

$$a = \left\lfloor \frac{a}{rs} \right\rfloor rs + k ; \quad b = \left\lfloor \frac{b}{rs} \right\rfloor rs + k$$

Taking mod r on both sides, we have

$$a \bmod r = \left(\left\lfloor \frac{a}{rs} \right\rfloor rs + k \right) \bmod r ; \quad b \bmod r = \left(\left\lfloor \frac{b}{rs} \right\rfloor rs + k \right) \bmod r$$

7 5 / 5

- 5 pts No submission
- ✓ - 0 pts Correct
- 3 pts Incorrect formula, not in closed form
- 2 pts Correct formula, not in closed form
- 1 pts Error
- 2 pts Incorrect closed formula

Simplifying further, we get the closed form expression for the sum of all integer multiples of x in $[\alpha, \beta]$:

$$S = \frac{\beta^2 - \alpha^2 + 2(\alpha \cdot \alpha \bmod x - \beta \cdot \beta \bmod x) + (\beta \bmod x)^2 - (\alpha \bmod x)^2}{2x} - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2}$$

Exercise 8. How many of the numbers 2^m , for $0 \leq m \leq M$ (where $m, M \in \mathbb{N}$), have leading digit 1 when written in decimal notation? Your answer should be a closed formula.

We first note that the number of digits in 2^m in decimal notation is given by $\lfloor \log_{10} 2^m \rfloor$. It is trivial to show that for any $M \in \mathbb{N}$, $0 \leq m := 0 \leq M$ and $2^0 = 1$ has leading digit 1. It is clear that each time there is a strict increase (necessarily by 1) in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$, the number of digits in 2^m increases by 1 and we have 2^m with leading digit 1 and vice-versa.

The number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ is therefore given by the number of unique terms in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$. Since the first term of this sequence is $\lfloor \log_{10} 2^1 \rfloor$ and the last term is $\lfloor \log_{10} 2^M \rfloor$, and each increase in the sequence is necessarily 1 and corresponds to a power of 2 that has leading digit 1, the number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ with leading digit 1 is given by

$$\lfloor \log_{10} 2^M \rfloor - \lfloor \log_{10} 2^1 \rfloor = \lfloor \log_{10} 2^M \rfloor - 0 = \lfloor \log_{10} 2^M \rfloor$$

Including the case where $m = 0$, we therefore have:

The number of terms in the sequence $(2^m)_{0 \leq m \leq M}$ that have leading digit 1 is given by $1 + \lfloor \log_{10} 2^M \rfloor$.

Exercise 9. Suppose $x, y, z \in \mathbb{Z}$ are such that $y, z \geq 1$. Prove that $z(x \bmod y) = (zx) \bmod(zy)$.

Define $k_1 := x \bmod y \in [0, y)$, $k_2 := (zx) \bmod(zy) \in [0, zy)$. By the definition of the mod operator, we have:

$$x = \left\lfloor \frac{x}{y} \right\rfloor y + k_1 ; \quad zx = \left\lfloor \frac{zx}{zy} \right\rfloor zy + k_2 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2$$

Plugging the expression for x from the first equation into the second, we get:

$$z \left(\left\lfloor \frac{x}{y} \right\rfloor y + k_1 \right) = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies \left\lfloor \frac{x}{y} \right\rfloor zy + zk_1 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies zk_1 = k_2$$

By definition of k_1 and k_2 , we now have $z(x \bmod y) = (zx) \bmod(zy)$. ■

Exercise 10. Suppose $a, b, r, s \in \mathbb{Z}$ are such that $r, s \geq 1$. Prove that if $a \bmod rs = b \bmod rs$, then $a \bmod r = b \bmod r$ and $a \bmod s = b \bmod s$.

Let $k := a \bmod(rs) = b \bmod(rs) \in [0, rs)$. By the definition of the module operator, we have

$$a = \left\lfloor \frac{a}{rs} \right\rfloor rs + k ; \quad b = \left\lfloor \frac{b}{rs} \right\rfloor rs + k$$

Taking mod r on both sides, we have

$$a \bmod r = \left(\left\lfloor \frac{a}{rs} \right\rfloor rs + k \right) \bmod r ; \quad b \bmod r = \left(\left\lfloor \frac{b}{rs} \right\rfloor rs + k \right) \bmod r$$

8 5 / 5

- 5 pts No submission
- ✓ - 0 pts Correct
- 1 pts Error
- 3 pts Incomplete
- 2 pts Incorrect argument
- 2 pts No final formula given
- 2 pts more justification needed

Simplifying further, we get the closed form expression for the sum of all integer multiples of x in $[\alpha, \beta]$:

$$S = \frac{\beta^2 - \alpha^2 + 2(\alpha \cdot \alpha \bmod x - \beta \cdot \beta \bmod x) + (\beta \bmod x)^2 - (\alpha \bmod x)^2}{2x} - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2}$$

Exercise 8. How many of the numbers 2^m , for $0 \leq m \leq M$ (where $m, M \in \mathbb{N}$), have leading digit 1 when written in decimal notation? Your answer should be a closed formula.

We first note that the number of digits in 2^m in decimal notation is given by $\lfloor \log_{10} 2^m \rfloor$. It is trivial to show that for any $M \in \mathbb{N}$, $0 \leq m := 0 \leq M$ and $2^0 = 1$ has leading digit 1. It is clear that each time there is a strict increase (necessarily by 1) in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$, the number of digits in 2^m increases by 1 and we have 2^m with leading digit 1 and vice-versa.

The number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ is therefore given by the number of unique terms in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$. Since the first term of this sequence is $\lfloor \log_{10} 2^1 \rfloor$ and the last term is $\lfloor \log_{10} 2^M \rfloor$, and each increase in the sequence is necessarily 1 and corresponds to a power of 2 that has leading digit 1, the number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ with leading digit 1 is given by

$$\lfloor \log_{10} 2^M \rfloor - \lfloor \log_{10} 2^1 \rfloor = \lfloor \log_{10} 2^M \rfloor - 0 = \lfloor \log_{10} 2^M \rfloor$$

Including the case where $m = 0$, we therefore have:

The number of terms in the sequence $(2^m)_{0 \leq m \leq M}$ that have leading digit 1 is given by $1 + \lfloor \log_{10} 2^M \rfloor$.

Exercise 9. Suppose $x, y, z \in \mathbb{Z}$ are such that $y, z \geq 1$. Prove that $z(x \bmod y) = (zx) \bmod(zy)$.

Define $k_1 := x \bmod y \in [0, y)$, $k_2 := (zx) \bmod(zy) \in [0, zy)$. By the definition of the mod operator, we have:

$$x = \left\lfloor \frac{x}{y} \right\rfloor y + k_1 ; \quad zx = \left\lfloor \frac{zx}{zy} \right\rfloor zy + k_2 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2$$

Plugging the expression for x from the first equation into the second, we get:

$$z \left(\left\lfloor \frac{x}{y} \right\rfloor y + k_1 \right) = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies \left\lfloor \frac{x}{y} \right\rfloor zy + zk_1 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies zk_1 = k_2$$

By definition of k_1 and k_2 , we now have $z(x \bmod y) = (zx) \bmod(zy)$. ■

Exercise 10. Suppose $a, b, r, s \in \mathbb{Z}$ are such that $r, s \geq 1$. Prove that if $a \bmod rs = b \bmod rs$, then $a \bmod r = b \bmod r$ and $a \bmod s = b \bmod s$.

Let $k := a \bmod(rs) = b \bmod(rs) \in [0, rs)$. By the definition of the module operator, we have

$$a = \left\lfloor \frac{a}{rs} \right\rfloor rs + k ; \quad b = \left\lfloor \frac{b}{rs} \right\rfloor rs + k$$

Taking mod r on both sides, we have

$$a \bmod r = \left(\left\lfloor \frac{a}{rs} \right\rfloor rs + k \right) \bmod r ; \quad b \bmod r = \left(\left\lfloor \frac{b}{rs} \right\rfloor rs + k \right) \bmod r$$

9 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Neglect to verify that $\leq z(x \bmod y) < zy$.

Simplifying further, we get the closed form expression for the sum of all integer multiples of x in $[\alpha, \beta]$:

$$S = \frac{\beta^2 - \alpha^2 + 2(\alpha \cdot \alpha \bmod x - \beta \cdot \beta \bmod x) + (\beta \bmod x)^2 - (\alpha \bmod x)^2}{2x} - \frac{((\alpha + \beta) - (\alpha \bmod x + \beta \bmod x))}{2}$$

Exercise 8. How many of the numbers 2^m , for $0 \leq m \leq M$ (where $m, M \in \mathbb{N}$), have leading digit 1 when written in decimal notation? Your answer should be a closed formula.

We first note that the number of digits in 2^m in decimal notation is given by $\lfloor \log_{10} 2^m \rfloor$. It is trivial to show that for any $M \in \mathbb{N}$, $0 \leq m := 0 \leq M$ and $2^0 = 1$ has leading digit 1. It is clear that each time there is a strict increase (necessarily by 1) in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$, the number of digits in 2^m increases by 1 and we have 2^m with leading digit 1 and vice-versa.

The number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ is therefore given by the number of unique terms in the sequence $(\lfloor \log_{10} 2^m \rfloor)_{1 \leq m \leq M}$. Since the first term of this sequence is $\lfloor \log_{10} 2^1 \rfloor$ and the last term is $\lfloor \log_{10} 2^M \rfloor$, and each increase in the sequence is necessarily 1 and corresponds to a power of 2 that has leading digit 1, the number of terms in the sequence $(2^m)_{1 \leq m \leq M}$ with leading digit 1 is given by

$$\lfloor \log_{10} 2^M \rfloor - \lfloor \log_{10} 2^1 \rfloor = \lfloor \log_{10} 2^M \rfloor - 0 = \lfloor \log_{10} 2^M \rfloor$$

Including the case where $m = 0$, we therefore have:

The number of terms in the sequence $(2^m)_{0 \leq m \leq M}$ that have leading digit 1 is given by $1 + \lfloor \log_{10} 2^M \rfloor$.

Exercise 9. Suppose $x, y, z \in \mathbb{Z}$ are such that $y, z \geq 1$. Prove that $z(x \bmod y) = (zx) \bmod(zy)$.

Define $k_1 := x \bmod y \in [0, y)$, $k_2 := (zx) \bmod(zy) \in [0, zy)$. By the definition of the mod operator, we have:

$$x = \left\lfloor \frac{x}{y} \right\rfloor y + k_1 ; \quad zx = \left\lfloor \frac{zx}{zy} \right\rfloor zy + k_2 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2$$

Plugging the expression for x from the first equation into the second, we get:

$$z \left(\left\lfloor \frac{x}{y} \right\rfloor y + k_1 \right) = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies \left\lfloor \frac{x}{y} \right\rfloor zy + zk_1 = \left\lfloor \frac{x}{y} \right\rfloor zy + k_2 \implies zk_1 = k_2$$

By definition of k_1 and k_2 , we now have $z(x \bmod y) = (zx) \bmod(zy)$. ■

Exercise 10. Suppose $a, b, r, s \in \mathbb{Z}$ are such that $r, s \geq 1$. Prove that if $a \bmod rs = b \bmod rs$, then $a \bmod r = b \bmod r$ and $a \bmod s = b \bmod s$.

Let $k := a \bmod(rs) = b \bmod(rs) \in [0, rs)$. By the definition of the module operator, we have

$$a = \left\lfloor \frac{a}{rs} \right\rfloor rs + k ; \quad b = \left\lfloor \frac{b}{rs} \right\rfloor rs + k$$

Taking mod r on both sides, we have

$$a \bmod r = \left(\left\lfloor \frac{a}{rs} \right\rfloor rs + k \right) \bmod r ; \quad b \bmod r = \left(\left\lfloor \frac{b}{rs} \right\rfloor rs + k \right) \bmod r$$

Since $k \in \mathbb{Z}$ and the mod operator is closed under addition for integer ② we then have ①

$$\begin{aligned} a \bmod r &= \left(\left\lfloor \frac{a}{rs} \right\rfloor rs \right) \bmod r + k \bmod r & b \bmod r &= \left(\left\lfloor \frac{b}{rs} \right\rfloor rs \right) \bmod r + k \bmod r \\ &= \left(\left\lfloor \frac{a}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r & &= \left(\left\lfloor \frac{b}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r \\ &= 0 + k \bmod r = k \bmod r & &= 0 + k \bmod r = k \bmod r \end{aligned}$$

We therefore have $a \bmod r = b \bmod r$. We can similarly show $a \bmod s = b \bmod s$. ■

Exercise 11. Suppose $b > 1$. Express $\log_b \log_b x$ in terms of $\ln \ln x$, $\ln \ln b$, and $\ln b$.

With simple logarithmic manipulations, we have:

$$\begin{aligned} \log_b \log_b x &= \log_b \frac{\ln x}{\ln b} && \text{(Change of base)} \\ &= \log_b \ln x - \log_b \ln b \\ &= \frac{\ln \ln x}{\ln b} - \frac{\ln \ln b}{\ln b} && \text{(Change of base)} \\ \therefore \log_b \log_b x &= \frac{\ln \ln x - \ln \ln b}{\ln b} \end{aligned}$$

Exercise 12 (Programming exercise, also doable by hand). If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000000.

Below is code, in C++, for a function that takes parameters `bound`, the upper bound for our search, and `factors`, a list of numbers at least one of which should factorise each number we add.

```

1 #include<vector>
2 using namespace std;
3
4 long long exercise12(int bound, vector<int> factors) {
5     // returns sum of all numbers under bound that are multiples
6     // of at least one factor in factors
7
8     long long sum = 0; // creating sum variable
9
10    // iterating over numbers under bound
11    for (int currentNum = 1; currentNum < bound; currentNum++) {
12        // iterating over factors
13        for (int factor : factors) {
14            // checking whether factor divides currentNum
15            if (currentNum % factor == 0) {
16                sum += currentNum; // updating sum
17                break; // breaking out of loop
18                // (don't need to check for other factors)
19            }
20        // returning sum
21        return sum;
22    }

```

Calling `exercise12(1000000, vector<int>{ 3, 5 })` returns 233333166668.

10 3 / 5

- 0 pts Correct
- 5 pts No submission
- 2 pts Assert $\$a \bmod r = a \bmod rs$.

✓ - 2 pts Assume that $\$x \mapsto x \bmod r$ is additive.

- 1 pts Missing step

1 It is not, e.g. $\$1+1 \bmod 2 = 0$.

2 "Closed under addition" is a property of a set, that if you add two numbers in the set you get a number in the set. The term you want it "additive", or "distributes over addition", or "commutes with addition".

Since $k \in \mathbb{Z}$ and the mod operator is closed under addition for integer ② we then have ①

$$\begin{aligned} a \bmod r &= \left(\left\lfloor \frac{a}{rs} \right\rfloor rs \right) \bmod r + k \bmod r & b \bmod r &= \left(\left\lfloor \frac{b}{rs} \right\rfloor rs \right) \bmod r + k \bmod r \\ &= \left(\left\lfloor \frac{a}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r & &= \left(\left\lfloor \frac{b}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r \\ &= 0 + k \bmod r = k \bmod r & &= 0 + k \bmod r = k \bmod r \end{aligned}$$

We therefore have $a \bmod r = b \bmod r$. We can similarly show $a \bmod s = b \bmod s$. ■

Exercise 11. Suppose $b > 1$. Express $\log_b \log_b x$ in terms of $\ln \ln x$, $\ln \ln b$, and $\ln b$.

With simple logarithmic manipulations, we have:

$$\begin{aligned} \log_b \log_b x &= \log_b \frac{\ln x}{\ln b} && \text{(Change of base)} \\ &= \log_b \ln x - \log_b \ln b \\ &= \frac{\ln \ln x}{\ln b} - \frac{\ln \ln b}{\ln b} && \text{(Change of base)} \\ \therefore \log_b \log_b x &= \frac{\ln \ln x - \ln \ln b}{\ln b} \end{aligned}$$

Exercise 12 (Programming exercise, also doable by hand). If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000000.

Below is code, in C++, for a function that takes parameters `bound`, the upper bound for our search, and `factors`, a list of numbers at least one of which should factorise each number we add.

```

1 #include<vector>
2 using namespace std;
3
4 long long exercise12(int bound, vector<int> factors) {
5     // returns sum of all numbers under bound that are multiples
6     // of at least one factor in factors
7
8     long long sum = 0; // creating sum variable
9
10    // iterating over numbers under bound
11    for (int currentNum = 1; currentNum < bound; currentNum++) {
12        // iterating over factors
13        for (int factor : factors) {
14            // checking whether factor divides currentNum
15            if (currentNum % factor == 0) {
16                sum += currentNum; // updating sum
17                break; // breaking out of loop
18                // (don't need to check for other factors)
19            }
20        // returning sum
21        return sum;
22    }

```

Calling `exercise12(1000000, vector<int>{ 3, 5 })` returns 233333166668.

11 5 / 5

- ✓ - 0 pts Correct.
- 5 pts No submission
- 2 pts Incorrect

Since $k \in \mathbb{Z}$ and the mod operator is closed under addition for integer ② we then have ①

$$\begin{aligned} a \bmod r &= \left(\left\lfloor \frac{a}{rs} \right\rfloor rs \right) \bmod r + k \bmod r & b \bmod r &= \left(\left\lfloor \frac{b}{rs} \right\rfloor rs \right) \bmod r + k \bmod r \\ &= \left(\left\lfloor \frac{a}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r & &= \left(\left\lfloor \frac{b}{rs} \right\rfloor s \cdot r \right) \bmod r + k \bmod r \\ &= 0 + k \bmod r = k \bmod r & &= 0 + k \bmod r = k \bmod r \end{aligned}$$

We therefore have $a \bmod r = b \bmod r$. We can similarly show $a \bmod s = b \bmod s$. ■

Exercise 11. Suppose $b > 1$. Express $\log_b \log_b x$ in terms of $\ln \ln x$, $\ln \ln b$, and $\ln b$.

With simple logarithmic manipulations, we have:

$$\begin{aligned} \log_b \log_b x &= \log_b \frac{\ln x}{\ln b} && \text{(Change of base)} \\ &= \log_b \ln x - \log_b \ln b \\ &= \frac{\ln \ln x}{\ln b} - \frac{\ln \ln b}{\ln b} && \text{(Change of base)} \\ \therefore \log_b \log_b x &= \frac{\ln \ln x - \ln \ln b}{\ln b} \end{aligned}$$

Exercise 12 (Programming exercise, also doable by hand). If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000000.

Below is code, in C++, for a function that takes parameters `bound`, the upper bound for our search, and `factors`, a list of numbers at least one of which should factorise each number we add.

```

1 #include<vector>
2 using namespace std;
3
4 long long exercise12(int bound, vector<int> factors) {
5     // returns sum of all numbers under bound that are multiples
6     // of at least one factor in factors
7
8     long long sum = 0; // creating sum variable
9
10    // iterating over numbers under bound
11    for (int currentNum = 1; currentNum < bound; currentNum++) {
12        // iterating over factors
13        for (int factor : factors) {
14            // checking whether factor divides currentNum
15            if (currentNum % factor == 0) {
16                sum += currentNum; // updating sum
17                break; // breaking out of loop
18                // (don't need to check for other factors)
19            }
20        // returning sum
21        return sum;
22    }

```

Calling `exercise12(1000000, vector<int>{ 3, 5 })` returns 233333166668.

12 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Program given without answer.

- 0 pts Wrong answer because of integer overflow

Exercise 13 (Programming exercise). Let F_0, F_1, F_2, \dots be the sequence of Fibonacci numbers. Compute the following sum:

$$\sum_{\substack{F_n < 10^7 \\ F_n \bmod 2 = 0}} F_n$$

Below is code, in C++, for a function that takes parameters `bound`, the upper bound for our search, and `factors`, a list of numbers at least one of which should factorise each Fibonacci number we add.

```

1 #include<vector>
2 using namespace std;
3
4 int exercise13(int bound, vector<int> factors) {
5     // returns the sum of all fibonacci numbers under bound that are divisible by
6     // any factor in factors
7
8     int sum = 0; // creating sum variable
9
10    int F_nm2 = 0; // F_(n-2)
11    int F_nm1 = 1; // F_(n-1)
12
13    while (true) {
14        // computing F_n
15        int F_n = F_nm2 + F_nm1;
16
17        // terminating condition
18        if (F_n >= bound)
19            return sum;
20
21        // checking whether F_n is divisible by any factor in factors
22        for (int factor : factors)
23            if (F_n % factor == 0) {
24                sum += F_n; // updating sum
25                break; // breaking out of loop (don't need to check for other factors)
26            }
27        // updating F_(n-2) and F_(n-1)
28        F_nm2 = F_nm1; F_nm1 = F_n;
29    }
30 }
```

Calling `exercise13(10000000, vector<int> { 2 })` returns 4613732.

13 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Fencepost error; added an extra Fibonacci number.

- 1 pts Incorrectly initialize sum.

Exercise 14 (Programming exercise). Determine the following number:

$$\min \{n \in \mathbb{N} : n \geq 1 \text{ and for each } k \in \{1, 2, \dots, 30\}, k|n\}$$

Note: the above number certainly exists since the above set contains the number $30!$, so it is non-empty and thus has a minimum element by the Well-Ordering Principle.

Below is code, in C++, for a function that takes the parameter `desiredFactors`, a list of all the numbers that must factorise our desired answer.

```
1 #include<vector>
2 using namespace std;
3
4 long long exercise14(vector<int> desiredFactors) {
5     // returns the smallest integer divisible by each number in desiredFactors
6     // desiredFactors = list of all numbers that should divide our desired number
7     // algorithm: first compute a list of the primes that themselves factorise all
8     // the desiredFactor
9     // then, iterating over desiredFactors, successively find the smallest number
10    // divisible by every factor encountered until that point
11
12    // note: this function doesn't work for values of n larger than 42 (of course it
13    //       's 42)
14    // this is not a limitation of the algorithm itself: the smallest number that
15    // is divisible by 1, 2, ..., 43
16    // is larger than long long can support (i.e. larger than 2^63)
17
18    // computing an upper-bound for the primes we need to consider
19    int primeBound = 0;
20    // iterating over factors to find largest factor
21    for (int factor : desiredFactors)
22        if (factor > primeBound)
23            primeBound = factor;
24
25    vector<int> primes;    // vector of all primes upto and including largest factor
26
27    // searching for primes (primitive algorithm) by iterating over the range (2,
28    // primeBound) (inclusive)
29    for (int i = 2; i <= primeBound; i++) {
30        bool primeFound = true;
31        // we assume we have a prime until and unless we find a factor
32        // iterating over the range(2, i - 1) (inclusive)
33        for (int j = 2; j < i; j++) {
34            // checking whether j divides i
35            if (i % j == 0) {
36                // j divides i -> i is not a prime
37                primeFound = false;
38                break; // don't need to continue checking for primality
39            }
40        }
41        // no factor of i found -> i is a prime
42        if (primeFound)
43            primes.push_back(i);
44    }
45}
```

```

41 long long currentMin = 1; // minimum number divisible by all numbers up to and
42     including desiredFactors[i]
43
44     for (int i = 0; i != desiredFactors.size(); i++) {
45
46         int n = desiredFactors[i];
47         // optimisation: if currentMin is already divisible by n, no need to multiply
48         or consider it
49         if (currentMin % n == 0)
50             continue;
51
52         long long nextMin = currentMin *= n;
53             // actual currentMin is now some factor of nextMin
54
55         // iterating over primes to continually divide candidateMin and get currentMin
56         for (int prime : primes) {
57             // dividing by current prime until dividing any further would violate the
58             loop invariant
59             while (true) {
60                 // checking whether candidateMin is divisible by prime
61                 if (nextMin % prime != 0)
62                     break;
63
64                 // computing next minCandidate
65                 long long minCandidate = nextMin / prime;
66                 // computing next candidate for min
67                 bool trueCandidate = true;
68                     // minCandidate is considered a true candidate by default
69
70                 // iterating over desiredFactors upto and including i'th index
71                 for (int j = 0; j <= i; j++)
72                     // if minCandidate is not divisible by a single desiredFactor,
73                     // it's not a valid candidate
74                     if (minCandidate % desiredFactors[j] != 0) {
75                         trueCandidate = false;
76                         break; // don't need to check further
77                     }
78
79                     // minCandidate is a true candidate -> update nextMin
80                     if (trueCandidate)
81                         nextMin = minCandidate;
82                     // minCandidate is not a true candidate
83                     // -> cannot divide by current prime further, break out of loop
84                     else break;
85                 }
86             // updating current minimum
87             currentMin = nextMin;
88         }
89         // return currentMin
90         return currentMin;
91     }

```

Calling `exercise14` with a list of the numbers from 1 through 30 gives 2329089562800.

14 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Give program without answer.

- 2 pts Serious error

Exercise 15 (Programming exercise). Let P_n denote the n th prime number. So $P_1 = 2, P_2 = 3, P_3 = 5, P_4 = 7, \dots$. Find P_{100000} .

Below is code, in C++, for a function that takes the parameter n and returns the n^{th} prime number.

```

1 #include<vector>
2 using namespace std;
3
4 int exercise15(int n) {
5     // returns the nth prime (primitive, brute force algorithm)
6
7     vector<int> primes = { 2 };    // initialising with 2 (so primes is non-empty)
8     int primesFound = 1;
9
10    // iterating from 3-onwards until we find nth prime
11    for (int primeCandidate = 3; true; primeCandidate++) {
12
13        bool primeTrue = true; // we assume primeCandidate is a prime by default
14        double primeCandidateRoot = sqrt(primeCandidate);
15
16        // iterating over primes and checking divisibility to check whether
17        primeCandidate is truly a prime
18        for (int prime : primes) {
19
20            // if prime > sqrt(primeCandidate), we don't need to check
21            // this is because the largest prime factor of any composite number is
22            less than or equal to its square root
23            if (prime > primeCandidateRoot)
24                break;
25
26            // prime divides prime candidate -> primeCandidate is not a prime
27            if (primeCandidate % prime == 0) {
28                primeTrue = false;
29                break; // don't need to check further
30            }
31
32            // primeCandidate is a prime -> add to primes, and increment primesFound
33            if (primeTrue) {
34                primes.push_back(primeCandidate);
35                primesFound++;
36            }
37            else continue;
38
39            // found nth prime -> return last prime found
40            if (primesFound == n)
41                return primes.back();
42        }
43    }
44 }
```

Calling `exercise15(100000)` returns 1299709.

15 5 / 5

- 5 pts No submission
- 2 pts Incorrect answer, needed to find $P_{\{100000\}}$ (one-hundred thousandth prime), not $P_{\{10000\}}$ (ten-thousandth prime)
- ✓ - 0 pts Correct
- 1 pts Error
- 2 pts Big error

Exercise 16 (Programming exercise). A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$$1/2 = 0.5, \quad 1/3 = 0.(3), \quad 1/4 = 0.25, \quad 1/5 = 0.2, \quad 1/6 = 0.1(6),$$

$$1/7 = 0.(142857), \quad 1/8 = 0.125, \quad 1/9 = 0.(1), \quad 1/10 = 0.1$$

where $0.1(6)$ means $0.166666\dots$, and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle. Find the value of $d < 3000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part. Hint: first analyze by hand what you would have to do to notice that $1/7$ has a 6-digit recurring cycle, think about it in terms of the Division Algorithm.

Below is code, in C++, of a function that takes parameter bound and returns integer $d \in [2, \text{bound})$ such that $1/d$ contains the longest recurring cycle in its decimal fraction part.

```

1 #include<unordered_map>
2 using namespace std;
3
4 int exercise16(int bound) {
5     // returns the number d in [2, bound) such that 1/d has the longest recurring
6     // cycle in its decimal fraction part
7
8     int currentBestD = 2;    // the current best value of d
9     int currentBestCycle = 0; // the current longest recurring cycle
10    // (naturally, associated with currentBestD)
11
12    // iterating over [2, bound)
13    // optimisation: larger values of d give longer cycles, so we start from the
14    // bound and decrement from there
15    for (int d = bound - 1; d >= 2; d--) {
16
17        // optimisation: 1/d can have a recurring cycle of length at most d-1
18        // if the currentBestCycle is larger than d-1, we don't need to check for
19        // the current and all lower values of d
20        if (currentBestCycle > d - 1) 4
21            break;
22
23        unordered_map<int, int> dividends;
24        // a map of the dividends obtained at each step of long division mapped
25        // with
26        // key = dividend of some step, value = index of digit in fractional part
27        // associated with the step
28        // unordered map gives O(1) search
29
30        int currentDividend = 1;      // current dividend in the long division algorithm
31        int currentStep = 0;         // current division step
32
33        // iterating until we find a recurring cycle / decimal expansion terminates
34        while (true) {
35
36            // inserting current dividend and step number
37            dividends.insert(pair<int, int> { currentDividend, currentStep++ });
38
39            // next dividend = remainder after current step of division
40            int nextDividend = (currentDividend * 10) % d;

```

```

36
37     // checking if division has terminated
38     if (nextDividend == 0) 3
39         break;
40
41     // division has not terminated -> checking if we have a cycle
42     bool recurringCycle = false;
43         // default assumption that there isn't a recurring cycle
44     int cycleLength = -1;
45         // default cycle length (modified only if there is a cycle)
46
47     // checking whether dividend has been previously find
48     unordered_map<int, int>::const_iterator itr = dividends.find(nextDividend);
49     if (itr != dividends.end()) {
50         recurringCycle = true;
51         // finding cycle length:
52             // currentStep - the step of the previous occurence of currentDividend
53         cycleLength = currentStep - itr->second;
54     }
55
56     // cycle found -> update currentBestD and currentBestCycle as necessary
57     if (recurringCycle) {
58         if (cycleLength > currentBestCycle) {
59             currentBestCycle = cycleLength;
60             currentBestD = d;
61         }
62         break; // break out of loop
63     }
64
65     // decimal expansion not terminated, cycle not found
66     // set up next iteration
67     currentDividend = nextDividend; // updating currentDividend
68 }
69 }
70 // return the best d found
71 return currentBestD;
72 }
```

Calling `exercise16(3000)` returns $d = 2971$, which has a recurring cycle of length 2970.

16 5 / 5

✓ - 0 pts Correct

- 5 pts No submission

- 1 pts Calculate number of distinct remainders instead of cycle length.

- 2 pts Incorrect

3 You don't really need to check whether division has terminated. A zero remainder produces a cycle of length 1.

4 Nice. I like this optimization.

Exercise 17. Suppose $f(n)$ and $g(n)$ are asymptotically positive functions. Prove that $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

We first show $f(n) = \Theta(g(n)) \implies f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

By definition of Ω -notation, there exist $c_1, c_2 \in \mathbb{R}$, $c_1, c_2 > 0$ and $n_0 \in \mathbb{N}$ such that

$$(1) \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

Let c_1, c_2 , and n_0 be as defined above. It is now easy to show $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

By definition of O -notation, $f(n) = O(g(n))$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$. Taking n_0 as defined above and $c = c_2$, from (1), we have

$$0 \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

We therefore have $f(n) = O(g(n))$.

By definition of Ω -notation, $f(n) = \Omega(g(n))$ if there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$. Taking n_0 as defined above and $c = c_1$, from (1), we have

$$0 \leq c_1 g(n) \leq f(n) \quad \forall n \geq n_0$$

We therefore have $f(n) = \Omega(g(n))$.

We now show $f(n) = O(g(n))$ and $f(n) = \Omega(g(n)) \implies f(n) = \Theta(g(n))$.

By definition of Ω -notation, there exist positive constants c_1 and n_1 such that $0 \leq c_1 g(n) \leq f(n)$ for all $n \geq n_1$. By definition of O -notation, there exist positive constants c_2 and n_2 such that $0 \leq f(n) \leq c_2 g(n)$ for all $n \geq n_2$.

Let $n_0 := \max\{n_0, n_1\}$. Combining these inequalities, we have

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

which is the necessary and sufficient condition to show $f(n) = \Theta(g(n))$.

$\therefore f(n) = \Theta(g(n)) \iff f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. ■

Exercise 18. Prove that for all $m \in \mathbb{N}$, $(\ln n)^m = o(n)$.

To prove $(\ln n)^m = o(n)$, it is sufficient to show $\lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} = 0$.

We assume $m \geq 1$ (the case where $m = 0$ is trivial, since $(\ln n)^0 = 1$ and the limit becomes $\lim_{n \rightarrow \infty} 1/n = 0$). Since $\ln n \rightarrow \infty$ as $n \rightarrow \infty$ and $m \geq 1$, $(\ln n)^m \rightarrow \infty$. We can therefore apply L'Hopital's Rule to get:

$$\lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} = \lim_{n \rightarrow \infty} \frac{m(\ln n)^{m-1}}{n}$$

Now, if $m - 1 = 0$, we again have the trivial case where the limit evaluates to 0. If $m - 1 > 0$, we can continue to apply L'Hopital's Rule until we get

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} &= \lim_{n \rightarrow \infty} \frac{m(\ln n)^{m-1}}{n} = \lim_{n \rightarrow \infty} \frac{m(m-1)(\ln n)^{m-2}}{n} \\ &= \dots = \lim_{n \rightarrow \infty} \frac{m!(\ln n)^0}{n} = \lim_{n \rightarrow \infty} \frac{m!}{n} = 0 \\ \therefore \lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} &= 0 \end{aligned}$$

$(\ln n)^m$ is therefore $o(n)$. ■

17 5 / 5

- 5 pts No submission
- ✓ - 0 pts Correct
- 1 pts Incomplete
- 2 pts Incomplete/Incorrect
- 2 pts Using asymptotic notation incorrectly

Exercise 17. Suppose $f(n)$ and $g(n)$ are asymptotically positive functions. Prove that $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

We first show $f(n) = \Theta(g(n)) \implies f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

By definition of Ω -notation, there exist $c_1, c_2 \in \mathbb{R}$, $c_1, c_2 > 0$ and $n_0 \in \mathbb{N}$ such that

$$(1) \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

Let c_1, c_2 , and n_0 be as defined above. It is now easy to show $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

By definition of O -notation, $f(n) = O(g(n))$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$. Taking n_0 as defined above and $c = c_2$, from (1), we have

$$0 \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

We therefore have $f(n) = O(g(n))$.

By definition of Ω -notation, $f(n) = \Omega(g(n))$ if there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$. Taking n_0 as defined above and $c = c_1$, from (1), we have

$$0 \leq c_1 g(n) \leq f(n) \quad \forall n \geq n_0$$

We therefore have $f(n) = \Omega(g(n))$.

We now show $f(n) = O(g(n))$ and $f(n) = \Omega(g(n)) \implies f(n) = \Theta(g(n))$.

By definition of Ω -notation, there exist positive constants c_1 and n_1 such that $0 \leq c_1 g(n) \leq f(n)$ for all $n \geq n_1$. By definition of O -notation, there exist positive constants c_2 and n_2 such that $0 \leq f(n) \leq c_2 g(n)$ for all $n \geq n_2$.

Let $n_0 := \max\{n_0, n_1\}$. Combining these inequalities, we have

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0$$

which is the necessary and sufficient condition to show $f(n) = \Theta(g(n))$.

$\therefore f(n) = \Theta(g(n)) \iff f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. ■

Exercise 18. Prove that for all $m \in \mathbb{N}$, $(\ln n)^m = o(n)$.

To prove $(\ln n)^m = o(n)$, it is sufficient to show $\lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} = 0$.

We assume $m \geq 1$ (the case where $m = 0$ is trivial, since $(\ln n)^0 = 1$ and the limit becomes $\lim_{n \rightarrow \infty} 1/n = 0$). Since $\ln n \rightarrow \infty$ as $n \rightarrow \infty$ and $m \geq 1$, $(\ln n)^m \rightarrow \infty$. We can therefore apply L'Hopital's Rule to get:

$$\lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} = \lim_{n \rightarrow \infty} \frac{m(\ln n)^{m-1}}{n}$$

Now, if $m - 1 = 0$, we again have the trivial case where the limit evaluates to 0. If $m - 1 > 0$, we can continue to apply L'Hopital's Rule until we get

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} &= \lim_{n \rightarrow \infty} \frac{m(\ln n)^{m-1}}{n} = \lim_{n \rightarrow \infty} \frac{m(m-1)(\ln n)^{m-2}}{n} \\ &= \dots = \lim_{n \rightarrow \infty} \frac{m!(\ln n)^0}{n} = \lim_{n \rightarrow \infty} \frac{m!}{n} = 0 \\ \therefore \lim_{n \rightarrow \infty} \frac{(\ln n)^m}{n} &= 0 \end{aligned}$$

$(\ln n)^m$ is therefore $o(n)$. ■

18 4 / 5

- 5 pts No submission
- 0 pts Correct
- ✓ - 1 pts Need to use induction, cannot include ... in a trail of equalities.
- 1 pts Circular reasoning
- 1 pts Error in argument
- 2 pts Incorrect use of log identity
- 3 pts No argument given
- 2 pts More rigor needed in argument