FARML Super Relay

Farmer John is sending a team of 15 cows to the Farm Animal Regional Mathematics League. As part of FARML, his cows will be compete in the infamous Super Relay event.

In this Super Relay, the contest moderator secretly selects a positive integer M and gives M to cows 1 and cows 15. Starting from both ends, the cows are instructed to solve a problem and pass the answer to the next cow in line. Cow 8 receives two answers and submits a final answer to the relay.

Submission format

The submission should be a single file `relay.py` (or any other file named `*.py`) which contains fifteen functions. Each function takes integer inputs and integer outputs. An example of a possible function is given below:

```
def f1(TNYWR):
    return TNYWR + 1
```

A template is provided at the end of this file for your convenience.

Gradescope uses **Python 3.6**.

Constraints

Each function will be called with integer inputs. It is guaranteed that each input is in $[0, 10000]$. Individual problems may have other implicit assumptions on their inputs for the task to be well-defined; it is guaranteed that all such assumptions hold.

Each function should return the answer as a function of the received number. It is guaranteed that each answer is an integer in $[0, 10000]$.

Scoring

Each cow's function is allocated one second of runtime.

A problem is solved if the answer to that problem is correct, *and* the answer to all problems that feed into it are correct. Thus, if cow 4 gets the wrong answer, then answers given by cows 5, 6, 7, 8 are not graded.

No partial credit is awarded within each problem. If the function throws an exception, it is graded as if an incorrect answer was given.

Leaderboard (for fun)

Just for fun, a leaderboard is shown with the submissions of other teams. The leaderboard shows the following metrics:

- Bonus score: sum of number of solves across thirty-six different choices of the value M, out of 540.
- The number of non-whitespace characters in the program's source file.
- The runtime of the program across solved problems, in microseconds.

The leaderboard is purely for amusement and is not needed for the puzzle.

Problems

R-1 Compute the sum of the divisors of 🐮.

R-2 Let S be the sum of the three smallest bases b for which, when 🐮 is written in base b, it becomes a palindrome. Compute $2S$.

R-3 Compute the 20th smallest positive integer coprime to 🐮.

R-4 Compute the largest prime less than or equal to 🐮 $+ 40$.

R-5 A class of 20 cows took an exam. Eight cows got the maximum score of 100. If the average of all scores is 🐮, compute the average of the scores of the cows who did not get a perfect score.

R-6 Compute the number m such that the remainder when 🐮 is divided by m is even, and the remainder is as large as possible.

R-7 There are $N/4$ distinct positive integers whose maximum is 🐮 and median is N. Compute the largest possible value of N.

R-15 A right triangle with integer side lengths $a < b < c$ has perimeter at most 🐮. Compute the maximum possible value of $ab - c$.

R-14 Let $f(n)$ denote the remainder when the sum of divisors of n^2 is divided by the sum of divisors of n. For example, $f(6) = 91 \bmod 12 = 7$. If k is the smallest positive integer for which $f(k) =$ 🐮, and it is given that $k \leq 10000$, find $f(k-1)$.

R-13 Each prime divisor of 🐮 is increased by 2 to obtain a new prime (with multiplicity). Compute the product of the new primes.

R-12 Compute the smallest integer greater than 🐮 which is a three-digit palindrome in some base. (Output your answer in base 10.)

R-11 A sequence x_1, x_2, ...of positive integers satisfies the recurrence $x_n = 2x_{n-1} + x_{n-3}$ for $n \geq 3$, where $x_0 = 0$ for convenience. Given that $x_7 \leq$ 🐮, compute the largest possible value of x_6.

R-10 Two positive integers have product 🐮. We increase the smaller integer by one, and decrease the larger integer by one. Compute the minimum possible value of the product of the two new integers.

R-9 A triangle-free simple graph G on 🐮 vertices has at least 🐮 edges. In addition, G has the property that adding any edge to it would form a triangle. Compute the minimum possible value of the number of edges.

R-8 Let $T =$ 🐮$_7$ and $S =$ 🐮$_9$. Find the smallest odd prime p such that

$$5^S + 5 \equiv T \pmod{p}.$$

Template file (download from `farml/relay.py`)

```python
# A few useful utility functions; you don't have to use them


def int_to_base(x: int, base: int) -> tuple[int, ...]:
    """Returns a tuple of base-b digits for integer x."""
    assert base >= 2 and x >= 0
    digits: list[int] = []
    while x:
        digits.append(int(x % base))
        x = int(x / base)
    return tuple(reversed(digits))


def get_primes(n: int) -> list[int]:
    """Returns a list of prime numbers < n."""
    # https://stackoverflow.com/a/3035188/4826845
    assert n > 0
    sieve = [True] * n
    for i in range(3, int(n**0.5) + 1, 2):
        if sieve[i]:
            sieve[i * i :: 2 * i] = [False] * ((n - i * i - 1) //
                (2 * i) + 1)
    return [2] + [i for i in range(3, n, 2) if sieve[i]]


# --- end ---


def f1(TNYWR):
    return -1


def f2(TNYWR):
    return -1


def f3(TNYWR):
    return -1


def f4(TNYWR):
    return -1


def f5(TNYWR):
    return -1


def f6(TNYWR):
    return -1


def f7(TNYWR):
    return -1


def f8(TNYWR7, TNYWR9):
    return -1


def f9(TNYWR):
    return -1


def f10(TNYWR):
```

```
    return -1


def f11(TNYWR):
    return -1


def f12(TNYWR):
    return -1


def f13(TNYWR):
    return -1


def f14(TNYWR):
    return -1


def f15(TNYWR):
    return -1
```