```python
def eval_palindrome(prime: int) -> bool:
  """
  Checks if the given prime is a palindrome arithmetically.
  Returns boolean.
  """

  if prime < 10:
    return True

  # saving prime to variable n to check if n == reverse
  n = prime

  # variable to track reversed prime value
  rev = 0

  # since we are using floor division for prime, we iterate until prime <= 0
  while prime > 0:

    # the currect digit being worked on is the remainder from mod 10, giving us the last digit
    dig = prime % 10

    # multiply current reverse by 10 to allow for addition of dig
    rev = (rev * 10) + dig

    # floor division on prime to truncate last digit
    prime = prime // 10

  # if n, the starting prime, is equal to the reverse then it is a palindrome
  if n == rev:
    return True
  else:
    return False
```

Algorithm to identify if the given number is palindromic.

A palindrome is defined as any string of characters which retains the exact same form or value when reversed.

The purpose of this algorithm was to identify palindromic prime numbers, and to see if there is a pattern or relationship amongst the palindromic primes. In order to efficiently identify these palindromes, we follow a series of simple steps.

The algorithm takes a single paramter, *prime*, which will be a prime number of any length.

First, we return True for any single-digit *prime*. By definition, any string composed of a single character will be palindromic.

Then, we begin to compose the reverse of *prime*. Since we cannot index individual digits of an integer and because we want this operation to be quick, we use modulo and floor division. We repeat this process until we have successfully built the reverse of *prime*.

Lastly, we compare *prime* to its reverse and return the result.

Our team looked at all prime numbers up to 10^9.