Competitive Programming

Numbers

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Prime Numbers

- Prime numbers are positive integers greater than 1 that have no divisors other than 1 and themselves.
- In other words, a prime number is a number that cannot be evenly divided by any other number except 1 and itself.
- For example, 2, 3, 5, 7, 11, 13, 17, and 19 are all prime numbers.

Time Complexity

- Check divisibility for numbers between 1 to n:
  - $O(n)$
- 1 to $\sqrt{n}$
  - $O(\sqrt{n})$

Sieve of Eratosthenes

- The Sieve of Eratosthenes is an ancient algorithm used to find all prime numbers up to a given limit.
- It was developed by the Greek mathematician Eratosthenes around 200 BC.
- The algorithm works by iteratively marking the multiples of each prime number, starting from 2, as composite (non-prime).

Steps

1. Create a list of consecutive integers from 2 to the desired limit.
2. Let the first number in the list be the current prime number (initially 2).
3. Starting from the current prime number, mark all its multiples as composite (not prime) by crossing them off the list.
4. Find the next number in the list that is not marked as composite, and set it as the new current prime number.
5. Repeat steps 3 and 4 until the square of the current prime number is greater than the limit.
6. All the remaining numbers in the list that are not marked as composite are prime numbers.

Example

- Start with a list of integers from 2 to 30.
- The first number is 2, which is prime. Cross off all its multiples: 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30.
- The next number not crossed off is 3. Cross off its multiples: 6, 9, 12, 15, 18, 21, 24, 27, 30.
- The next number not crossed off is 5. Cross off its multiples: 10, 15, 20, 25, 30.
- The next number not crossed off is 7. Cross off its multiples: 14, 21, 28.
- We stop here because the square of the next number (7^2 = 49) is greater than the limit (30).
- The remaining numbers that are not crossed off are prime: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.

Generate all prime no. between 1 to 50

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Generate all prime no. between 1 to 50

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Generate all prime no. between 1 to 50

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Generate all prime no. between 1 to 50

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Consecutive Prime Sum

- Some prime numbers can be expressed as Sum of other consecutive prime numbers.
- For example
  - 5 = 2 + 3
  - 17 = 2 + 3 + 5 + 7
  - 41 = 2 + 3 + 5 + 7 + 11 + 13
- Your task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.
- Write code to find out number of prime numbers that satisfy the above mentioned property in a given range.

Sample input and output

Input

2

20

15

Output

2

1

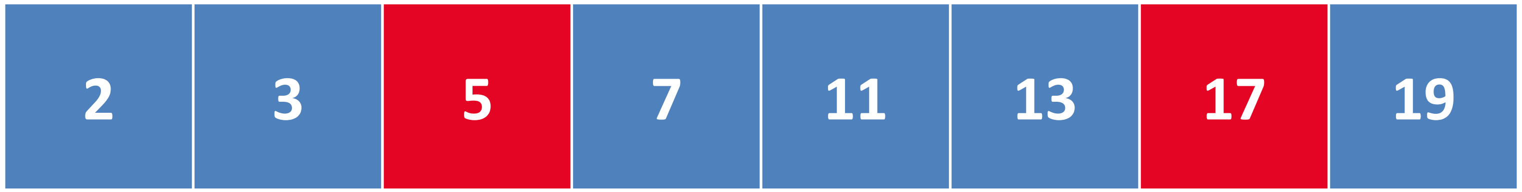
Example

Input

20

Output

2



Approach

- Built list of prime numbers and store in an array.
- For each prime, check if it’s sum of previous primes.

Pseudo code: Generating list

```
count = 0
for i = 2 to n + 1:
        if isPrime(i):
                primesList[count] = i
                count++
```

Pseudo code: isPrime

```
for i = 2 to sqrt(n):
    if n % i == 0:
        return false
return true
```

Pseudo code: Checking sum

```
primesCount = 0
for i = 1 to count - 1:
    sum = 0
    for j = 0 to i - 1:
        sum = sum + primesList[j]
        if sum == primesList[i]:
            primesCount++
            break
print primesCount
```

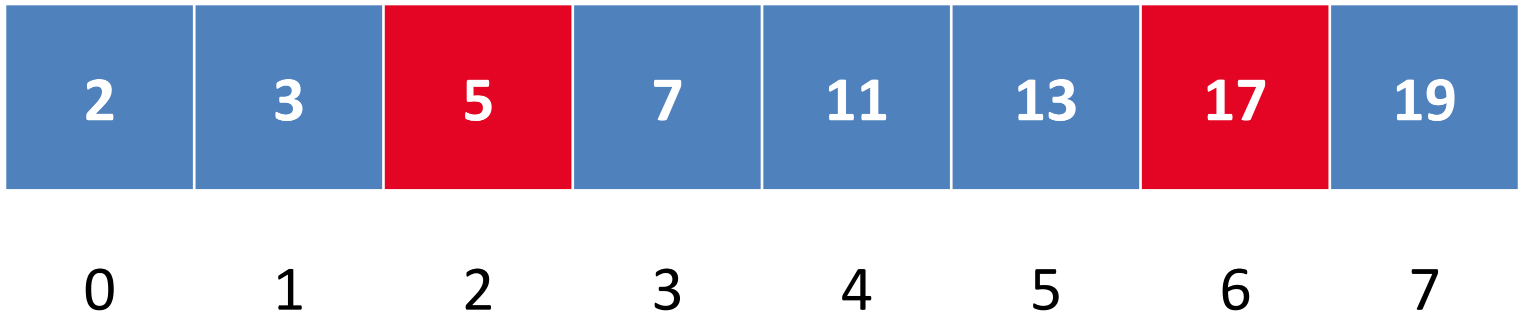
Example

Input

20

Output

2



Concatenating Primes

- If you like numbers, you may have been fascinated by prime numbers.
- Sometimes we obtain by concatenating two primes.
- For example, concatenating 2 and 3, we obtain the prime 23.
- The aim is to find all such distinct "concatenated primes" that could be obtained by concatenating primes ≤ a given integer N.

Input and output format

- Input Format:
  - Integer N
- Output Format:
  - M, the number of distinct primes that could be obtained by concatenating two primes ≤ N.

Constraints

- $N \leq 70$

Sample input and output

Input

10

Output

4

Explanation

- The primes ≤ 10 are 2, 3, 5, 7.
- These can be used to form the following concatenated numbers: 22, 23, 25, 27, 32, 33, 35, 37, 52, 53, 55, 57, 72, 73, 75, 77.
- Of these, there are four primes: 23 37 53 and 73.
- Hence the output is 4.

Sample input and output

Input

20

Output

17

Explanation

- The prime numbers up to 20 are 2 3 5 7 11 13 17 and 19.
- Concatenating these two at a time in all possible ways, we get the following numbers:
    - 22 23 25 27 211 213 217 219
    - 32 33 35 37 311 313 317 319
    - 52 53 55 57 511 513 517 519
    - 72 73 75 77 711 713 717 719
    - 112 113 115 117 1111 1113 1117 1119
    - 132 133 135 137 1311 1313 1317 1319
    - 172 173 175 177 1711 1713 1717 1719
    - 192 193 195 197 1911 1913 1917 1919

Explanation

- We have the following 17 primes numbers in this list:
    - 23 37 53 73 113 137 173 193 197 211 311 313 317 719 1117 1319 1913
- Hence the output would be 17.

Pseudo code

```
input n
for i = 2 to n:
    if isPrime(i):
        primes[primesCount++] = i
for i = 0 to primesCount - 1:
    for j = 0 to primesCount - 1:
        if isPrime(concatenate(primes[i], primes[j])):
            count++
print count
```

Pseudo code: isPrime(n)

```
for i = 2 to (int) sqrt(n):
        if n % i == 0:
                return 0
return 1
```

Pseudo code: concatenate(a, b)

```
if b < 10:
        return a * 10 + b
else:
        return a * 100 + b
```

Game of Primes

- In a global Mathematics contest, the contestants are told to invent some special numbers which can be built by adding the squares of its digits.
- Doing this perpetually, the numbers will end up to 1 or 4.
- If a positive integer ends with 1, then it is called the Number of Game.
- An example from above is:
  - 13 = 1^2 + 3^2 = 1+9 = 10 (Step:1)
  - 10 = 1^2 + 0^2 = 1+0 = 1 (Step:2), iteration ends in Step 2 since number ends with 1
- Then in next round, the contestants are asked to combine their newly invented number, i.e. Number of Game with prime number.

Game of Primes

- Now, being a smart programmer, write a program to help the contestants to find out the N^{th} combined number within any given range, where N can be any integer.

Sample input and output

Input

```
1
30
3
```

Output

```
19
```

Sample input and output

Input

12

33

5

Output

No number is present at this index

Sample input and output

Input

```
-5
@
4
```

Output

```
Invalid Input
```

Questions?

- What is a number of game?
- What is a prime number?

Example

- 7
- $7^2 \rightarrow 49$
- $4^2 + 9^2 \rightarrow 16 + 81 \rightarrow 97$
- $9^2 + 7^2 \rightarrow 81 + 49 \rightarrow 130$
- $1^2 + 3^2 + 0^2 \rightarrow 10$
- $1^2 + 0^2 \rightarrow 1$

Squared digit sum

- Take any positive integer *n* and sum the squares of its digits.
- If you repeat this operation, eventually you'll either end at 1 or cycle between the eight values 4, 16, 37, 58, 89, 145, 42, and 20.
- For example, pick $n = 389$. Then $3^2 + 8^2 + 9^2 = 9 + 64 + 81 = 154$.
- Next, $1^2 + 5^2 + 4^2 = 1 + 25 + 16 = 42$, and now we're at one of the eight special values.
- You can easily verify that once you land on one of these values you keep cycling.

$$\begin{tikzcd}
4 \arrow[r] & 16 \arrow[r] & 37 \arrow[r] & 58 \arrow[d] \\
20 \arrow[u] & 42 \arrow[l] & 145 \arrow[l] & 89 \arrow[l]
\end{tikzcd}$$

Happy numbers (up to 1000)

1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, 103, 109, 129, 130, 133, 139, 167, 176, 188, 190, 192, 193, 203, 208, 219, 226, 230, 236, 239, 262, 263, 280, 291, 293, 301, 302, 310, 313, 319, 320, 326, 329, 331, 338, 356, 362, 365, 367, 368, 376, 379, 383, 386, 391, 392, 397, 404, 409, 440, 446, 464, 469, 478, 487, 490, 496, 536, 556, 563, 565, 566, 608, 617, 622, 623, 632, 635, 637, 638, 644, 649, 653, 655, 656, 665, 671, 673, 680, 683, 694, 700, 709, 716, 736, 739, 748, 761, 763, 784, 790, 793, 802, 806, 818, 820, 833, 836, 847, 860, 863, 874, 881, 888, 899, 901, 904, 907, 910, 912, 913, 921, 923, 931, 932, 937, 940, 946, 964, 970, 973, 989, 998, 1000

Prime numbers (up to 100)

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997

Nos. which are both Happy & Prime

7, 13, 19, 23, 31, 79, 97, 103, 109, 139, 167, 193, 239, 263, 293, 313, 331, 367, 379, 383, 397, 409, 487, . . .

Approach

- Input numbers X, Y and N
- i from X to Y
  - Check if i isPrime and isHappy
    - Then increment the Count
  - If Count == N, the print i
- Invalid Input / No number is present at this index

Pseudo code: isHappy

```
sum = 0
while n > 0:
        sum = sum + (n % 10) * (n % 10);
        n = n / 10;
if sum == 1:
        return true
else if sum == 4:
        return false
return isHappy(sum)
```

Pseudo code: isPrime

```
if n == 1:
    return false
for i = 2 to sqrt(n):
    if n % i == 0:
        return false
return true
```

Queries?

Thank You…!