Experiment 8

Dynamic programming

Student Name: Bhanu Pundir **UID:** 20BCS1439
Branch: BE CSE **Section/Group:** 62-B
Semester: 5th **Date of Performance:** 04 Nov 2022
Subject Name: CC Lab **Subject Code:** 20CSP-314

1. **Aim/Overview of the practical:**

Dynamic Programming

Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

https://www.hackerrank.com/challenges/sam-and-substrings/problem?isFullScreen=true

2. **Apparatus / Simulator Used:**
   - Windows 7 or above
   - Google Chrome

3. **Objective:**
   - To understand the concept of Dynamic Programming.
   - To implement the concept of Dynamic Programming.
   - Goal is to find the number of ways to construct an array such that consecutive positions contain different values.

4. **Code:**

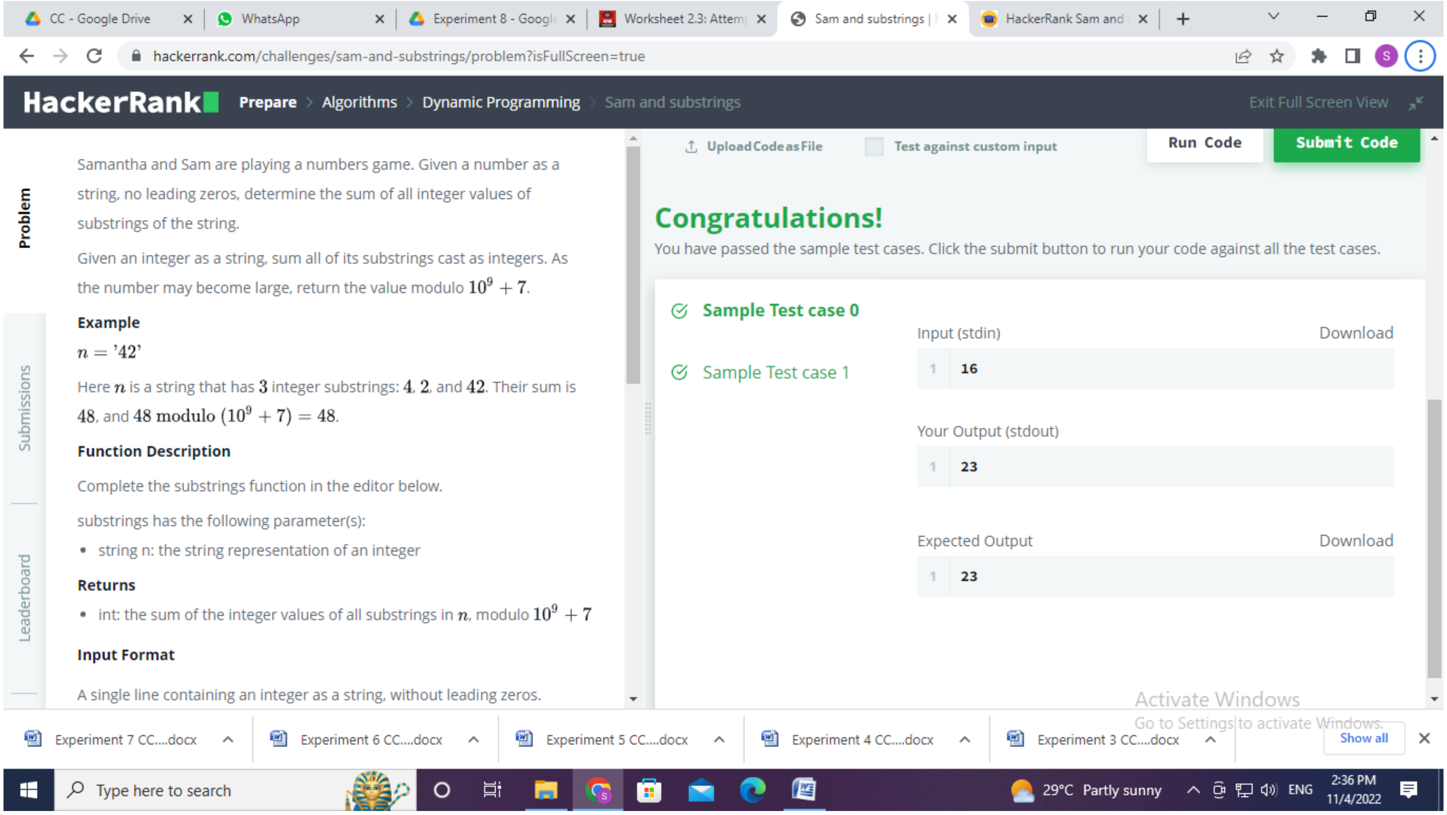
```
def solution(n):
    s = 0
    prev_sum = 0

    for i, d in enumerate(n):
        s_ = prev_sum * 10 + (i + 1) * int(d)
        s += s_
        prev_sum = s_
    return s % (10 ** 9 + 7)

n = input()

print(solution(n))
```

5. Result/Output/Writing Summary:



Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Example

$n = '42'$

Here n is a string that has 3 integer substrings: 4, 2, and 42. Their sum is 48, and $48 \text{ modulo } (10^9 + 7) = 48$.

Function Description

Complete the substrings function in the editor below.

substrings has the following parameter(s):

- string n: the string representation of an integer

Returns

- int: the sum of the integer values of all substrings in n, modulo $10^9 + 7$

Input Format

A single line containing an integer as a string, without leading zeros.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

Input (stdin) Download

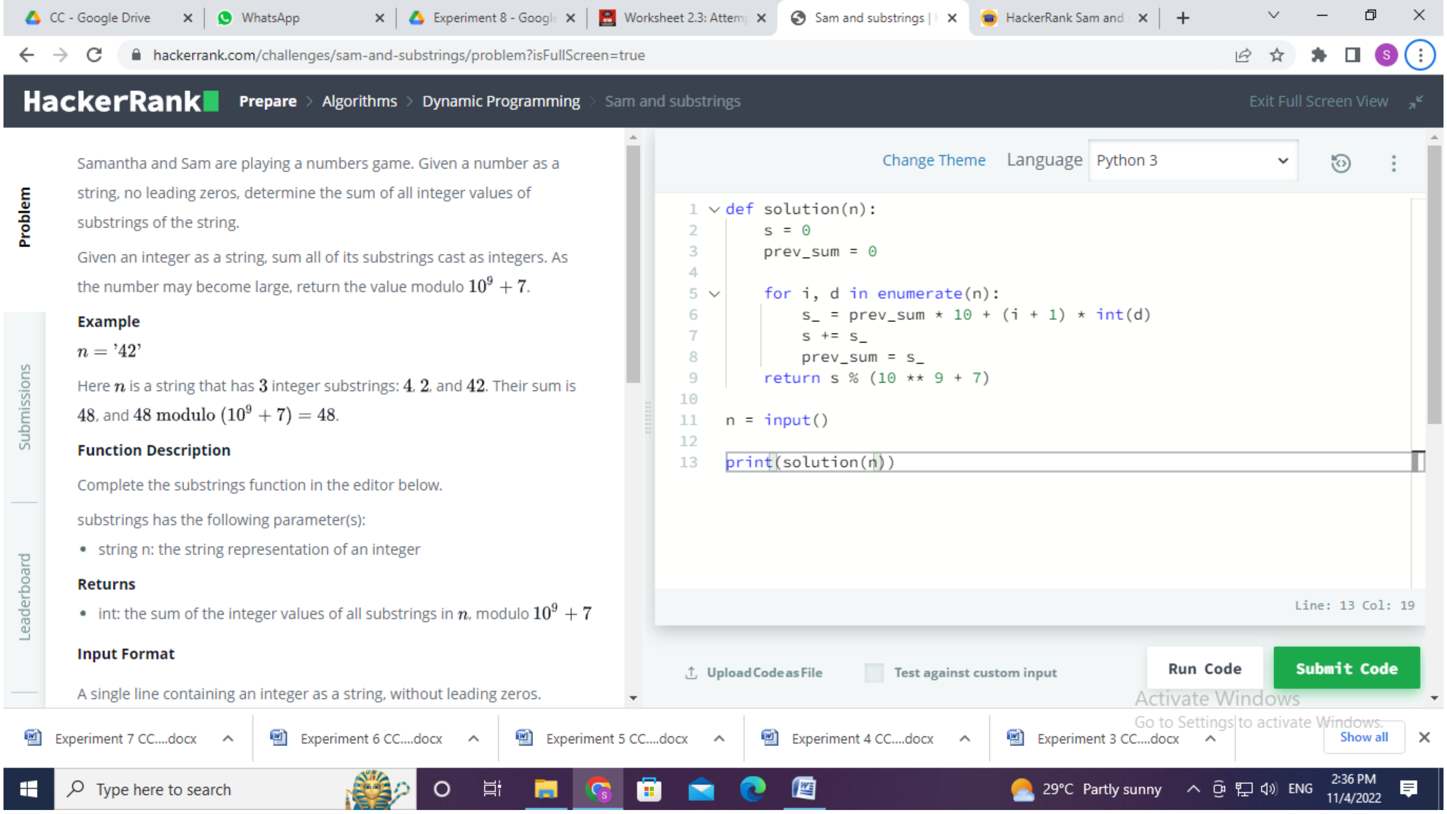
```
16
```

Your Output (stdout)

```
23
```

Expected Output Download

```
23
```



Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Example

$n = \text{'42'}$

Here n is a string that has 3 integer substrings: 4, 2, and 42. Their sum is 48, and $48 \text{ modulo } (10^9 + 7) = 48$.

Function Description

Complete the substrings function in the editor below.

substrings has the following parameter(s):

- string n: the string representation of an integer

Returns

- int: the sum of the integer values of all substrings in n, modulo $10^9 + 7$

Input Format

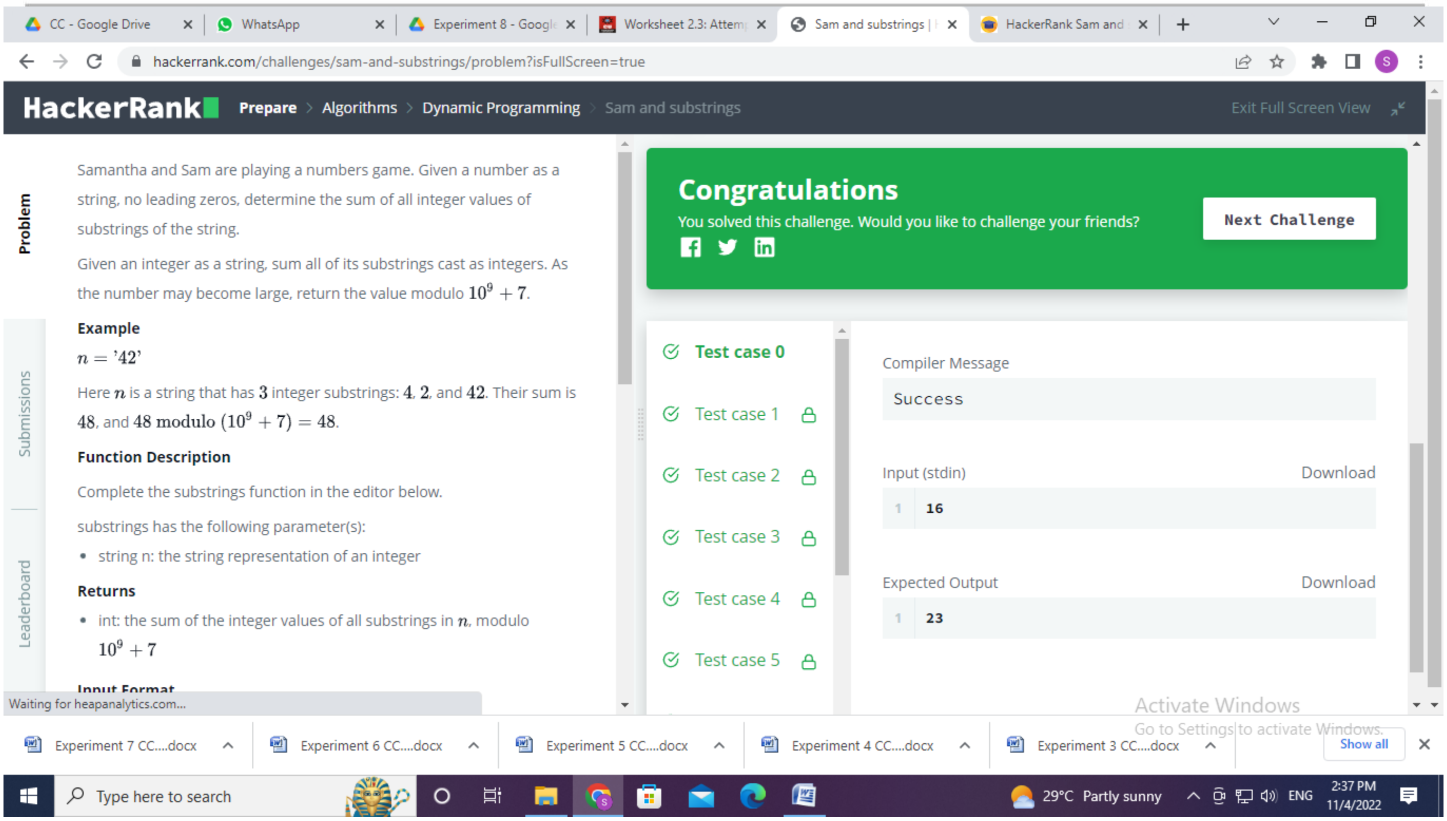
A single line containing an integer as a string, without leading zeros.

```python
def solution(n):
    s = 0
    prev_sum = 0

    for i, d in enumerate(n):
        s_ = prev_sum * 10 + (i + 1) * int(d)
        s += s_
        prev_sum = s_
    return s % (10 ** 9 + 7)

n = input()

print(solution(n))
```



Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Example

$n = '42'$

Here n is a string that has 3 integer substrings: 4, 2, and 42. Their sum is 48, and $48 \text{ modulo } (10^9 + 7) = 48$.

Function Description

Complete the substrings function in the editor below.

substrings has the following parameter(s):

- string n: the string representation of an integer

Returns

- int: the sum of the integer values of all substrings in n, modulo $10^9 + 7$

Input Format

Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

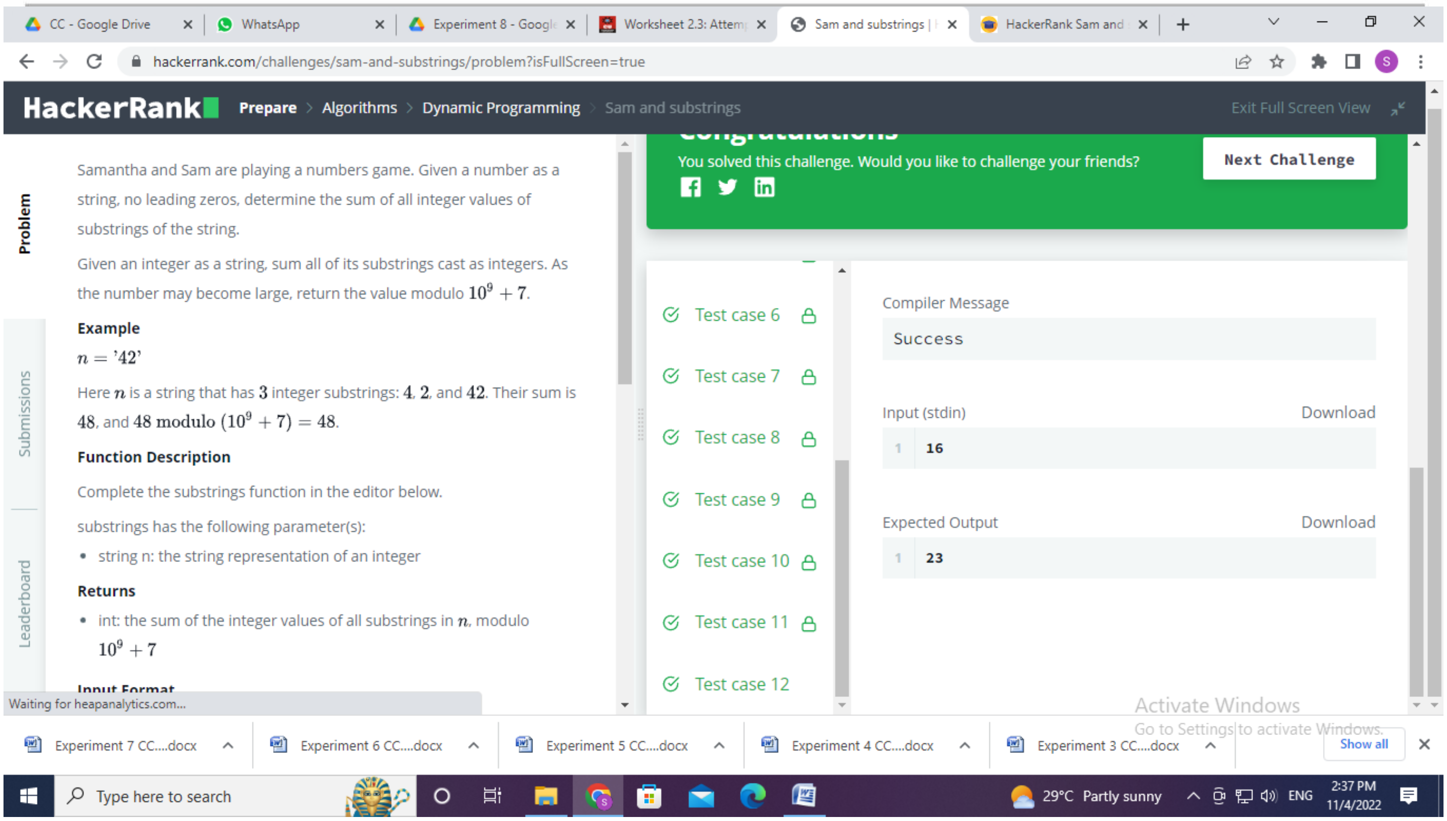
```
Success
```

Input (stdin)

```
16
```

Expected Output

```
23
```



Samantha and Sam are playing a numbers game. Given a number as a string, no leading zeros, determine the sum of all integer values of substrings of the string.

Given an integer as a string, sum all of its substrings cast as integers. As the number may become large, return the value modulo $10^9 + 7$.

Example

$n = '42'$

Here n is a string that has 3 integer substrings: 4, 2, and 42. Their sum is 48, and $48 \text{ modulo } (10^9 + 7) = 48$.

Function Description

Complete the substrings function in the editor below.

substrings has the following parameter(s):

- string n: the string representation of an integer

Returns

- int: the sum of the integer values of all substrings in n, modulo $10^9 + 7$

Input Format

You solved this challenge. Would you like to challenge your friends?

✓ Test case 6

✓ Test case 7

✓ Test case 8

✓ Test case 9

✓ Test case 10

✓ Test case 11

✓ Test case 12

Compiler Message

```
Success
```

Input (stdin) Download

```
16
```

Expected Output Download

```
23
```

Experiment 8.2

1. Aim/Overview of the practical:

DynamicProgramming

Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in.

https://www.hackerrank.com/challenges/red-john-is-back/problem?isFullScreen=true

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Dynamic Programming.
- To implement the concept of Dynamic Programming.
- Goal is to find the number of ways to construct an array such that consecutive positions contain different values.

4. Code:

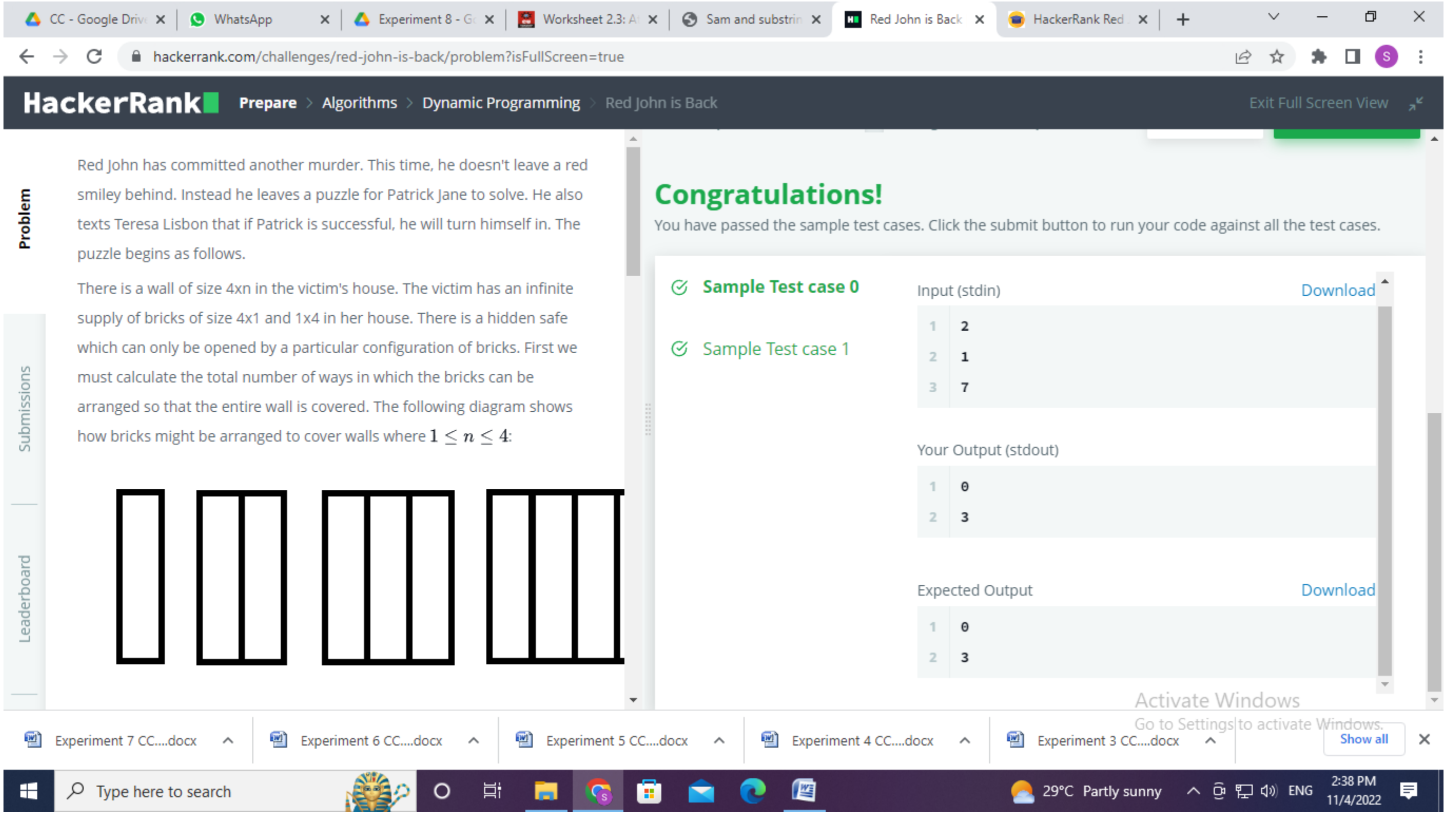
```
def primes(n):
    """ Returns  a list of primes < n """
    if n <= 2: return 0
    sieve = [True] * n
    for i in range(3,int(n**0.5)+1,2):
        if sieve[i]:
            sieve[i*i::2*i]=[False]*int((n-i*i-1)/(2*i)+1)
    return len([i for i in range(3,n,2) if sieve[i]]) + 1

def find_configs(N):
    if N == 0:
        return 1
    elif N < 0:
        return 0

    return find_configs(N-1) + find_configs(N-4)

T = int(input())
for i in range(T):
    print(primes(find_configs(int(input()))+1))
```

5. Result/Output/Writing Summary:



Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in. The puzzle begins as follows.

There is a wall of size 4xn in the victim's house. The victim has an infinite supply of bricks of size 4x1 and 1x4 in her house. There is a hidden safe which can only be opened by a particular configuration of bricks. First we must calculate the total number of ways in which the bricks can be arranged so that the entire wall is covered. The following diagram shows how bricks might be arranged to cover walls where $1 \leq n \leq 4$:

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ **Sample Test case 0**

✓ Sample Test case 1

Input (stdin)

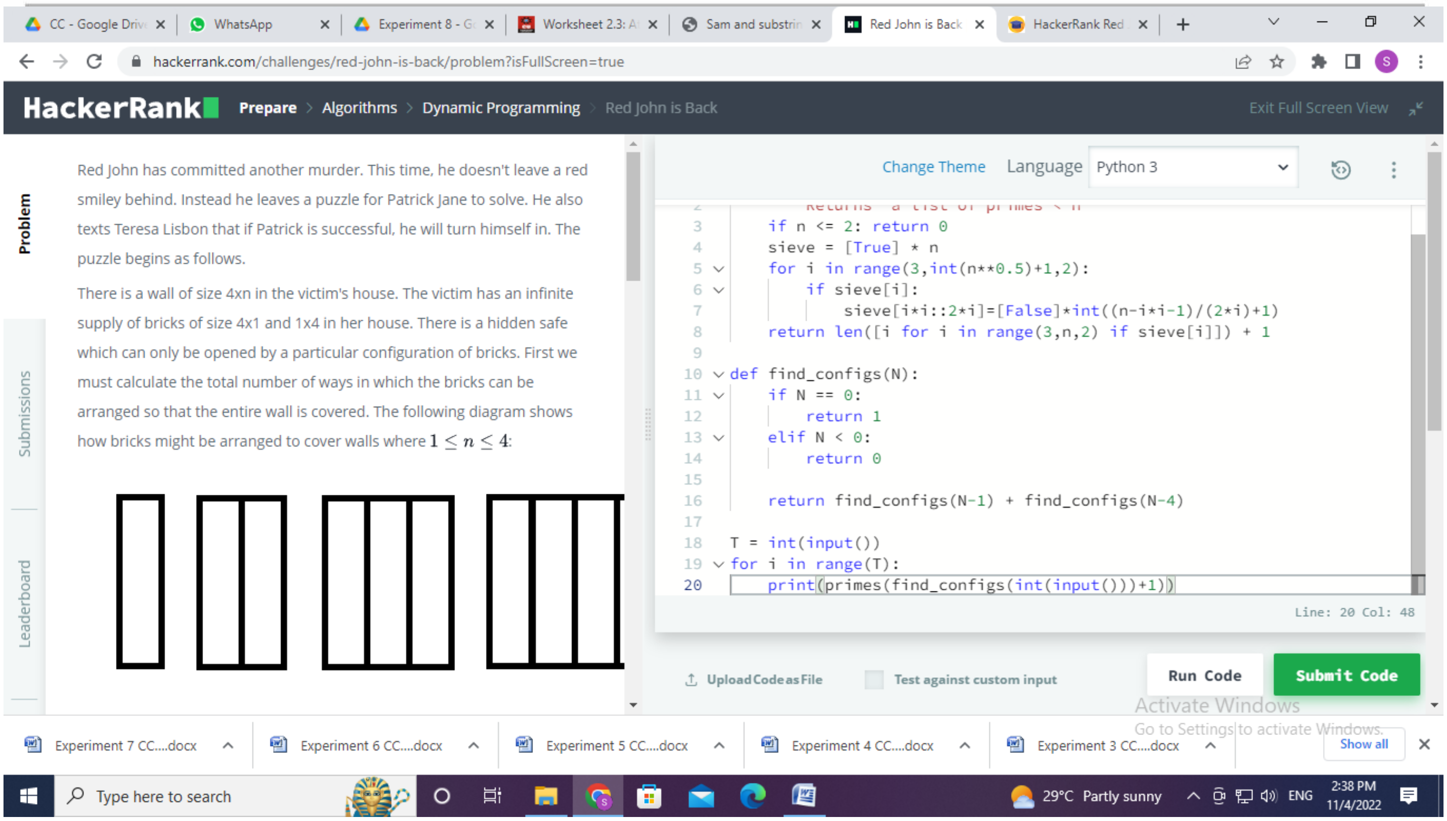
```
2
1
7
```

Your Output (stdout)

```
0
3
```

Expected Output

```
0
3
```



Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in. The puzzle begins as follows.

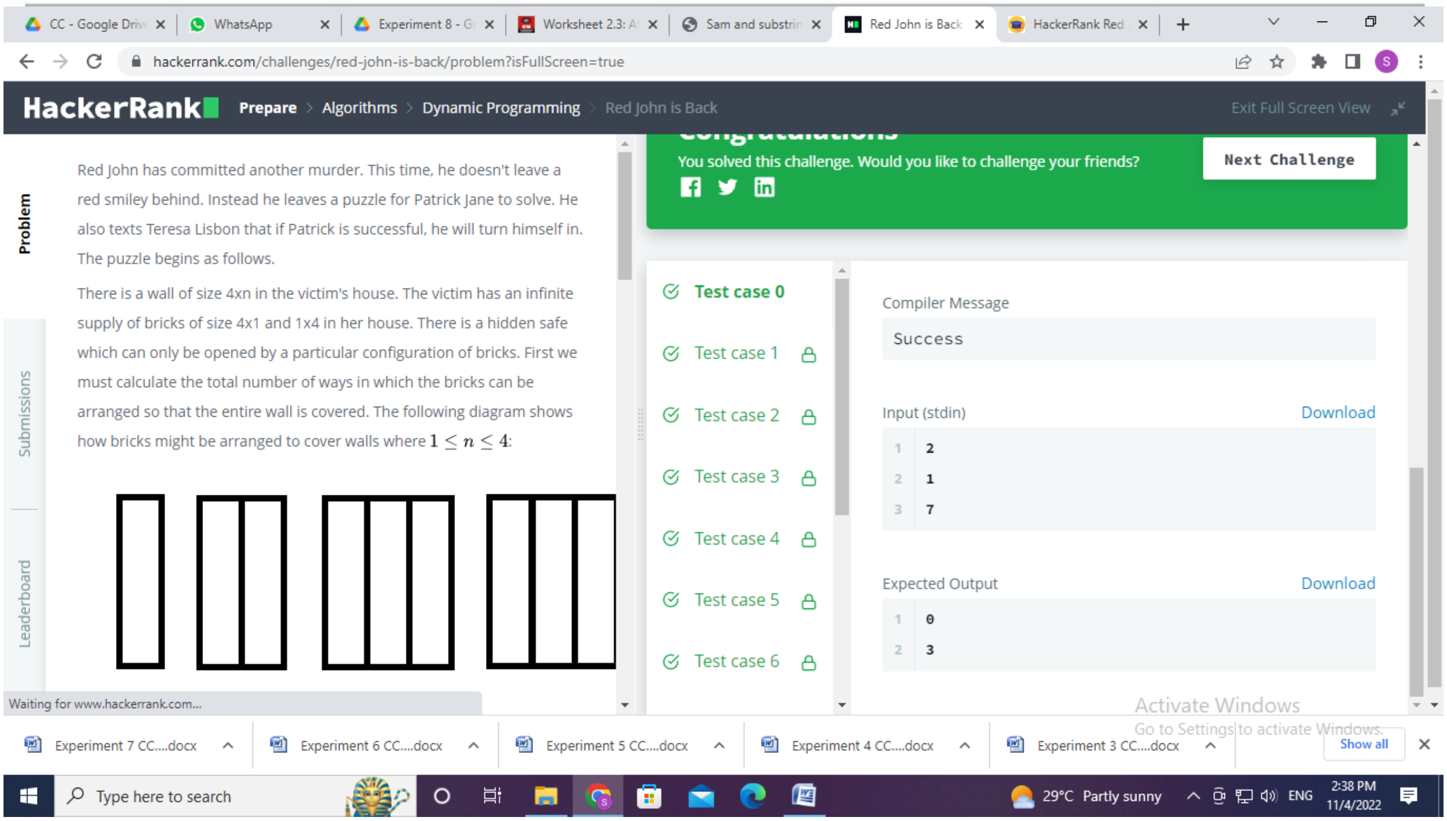
There is a wall of size 4xn in the victim's house. The victim has an infinite supply of bricks of size 4x1 and 1x4 in her house. There is a hidden safe which can only be opened by a particular configuration of bricks. First we must calculate the total number of ways in which the bricks can be arranged so that the entire wall is covered. The following diagram shows how bricks might be arranged to cover walls where $1 \leq n \leq 4$:

```
        Returns  a list of primes < n
    if n <= 2: return 0
    sieve = [True] * n
    for i in range(3,int(n**0.5)+1,2):
        if sieve[i]:
            sieve[i*i::2*i]=[False]*int((n-i*i-1)/(2*i)+1)
    return len([i for i in range(3,n,2) if sieve[i]]) + 1

def find_configs(N):
    if N == 0:
        return 1
    elif N < 0:
        return 0

    return find_configs(N-1) + find_configs(N-4)

T = int(input())
for i in range(T):
    print(primes(find_configs(int(input()))+1))
```



Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in. The puzzle begins as follows.

There is a wall of size 4xn in the victim's house. The victim has an infinite supply of bricks of size 4x1 and 1x4 in her house. There is a hidden safe which can only be opened by a particular configuration of bricks. First we must calculate the total number of ways in which the bricks can be arranged so that the entire wall is covered. The following diagram shows how bricks might be arranged to cover walls where $1 \le n \le 4$:

Waiting for www.hackerrank.com...

You solved this challenge. Would you like to challenge your friends?

✓ Test case 0
✓ Test case 1
✓ Test case 2
✓ Test case 3
✓ Test case 4
✓ Test case 5
✓ Test case 6

Compiler Message

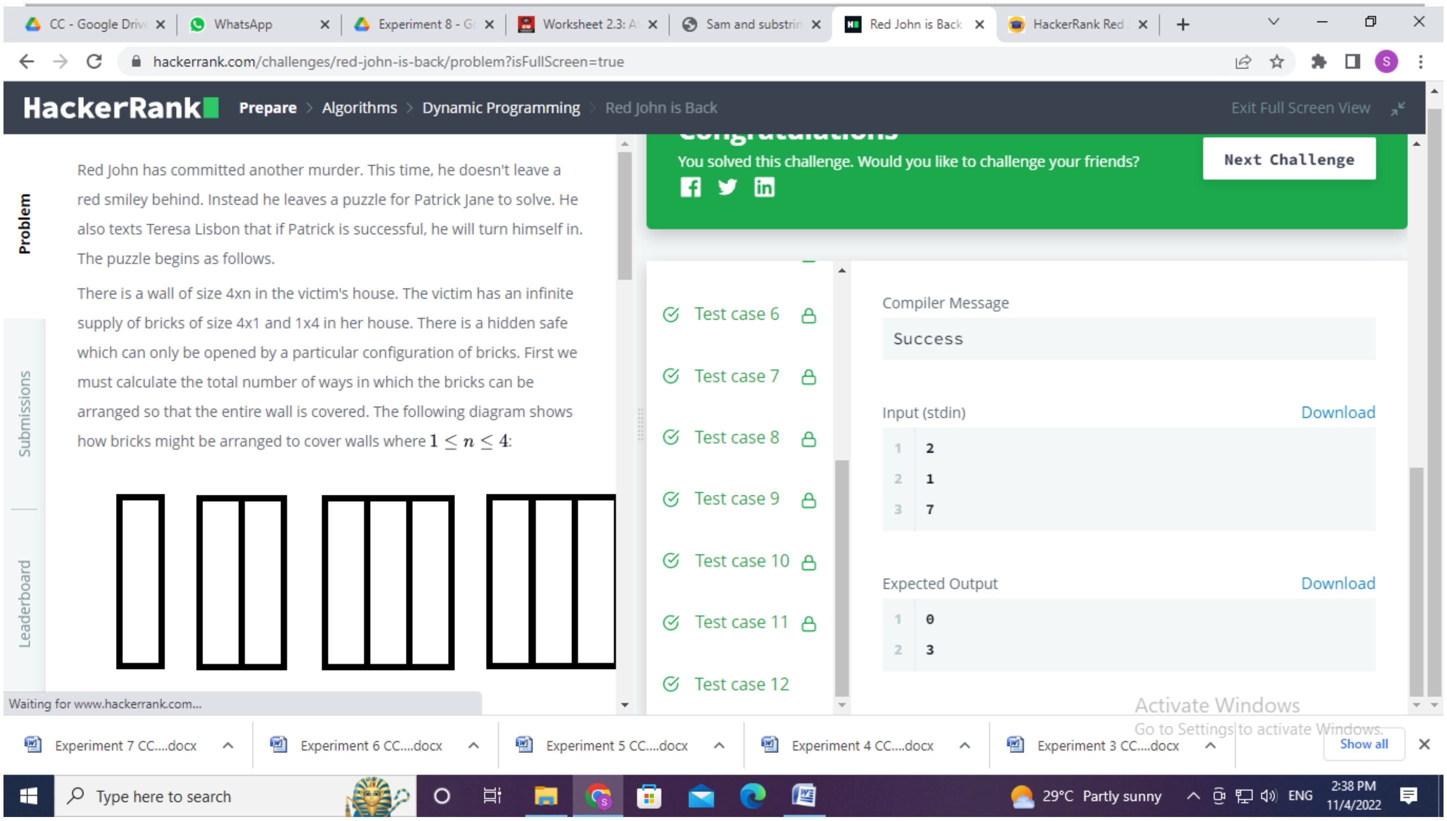
```
Success
```

Input (stdin) Download

```
2
1
7
```

Expected Output Download

```
0
3
```



Red John has committed another murder. This time, he doesn't leave a red smiley behind. Instead he leaves a puzzle for Patrick Jane to solve. He also texts Teresa Lisbon that if Patrick is successful, he will turn himself in. The puzzle begins as follows.

There is a wall of size 4xn in the victim's house. The victim has an infinite supply of bricks of size 4x1 and 1x4 in her house. There is a hidden safe which can only be opened by a particular configuration of bricks. First we must calculate the total number of ways in which the bricks can be arranged so that the entire wall is covered. The following diagram shows how bricks might be arranged to cover walls where $1 \leq n \leq 4$:

You solved this challenge. Would you like to challenge your friends?

- Test case 6
- Test case 7
- Test case 8
- Test case 9
- Test case 10
- Test case 11
- Test case 12

Compiler Message

```
Success
```

Input (stdin)

```
2
1
7
```

Expected Output

```
0
3
```

Learning outcomes (What I have learnt):

- Learned the concept of Dynamic Programming.
- Learned about Array in Dynamic Programming.
- Learn about the countArray and Equal concept.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			