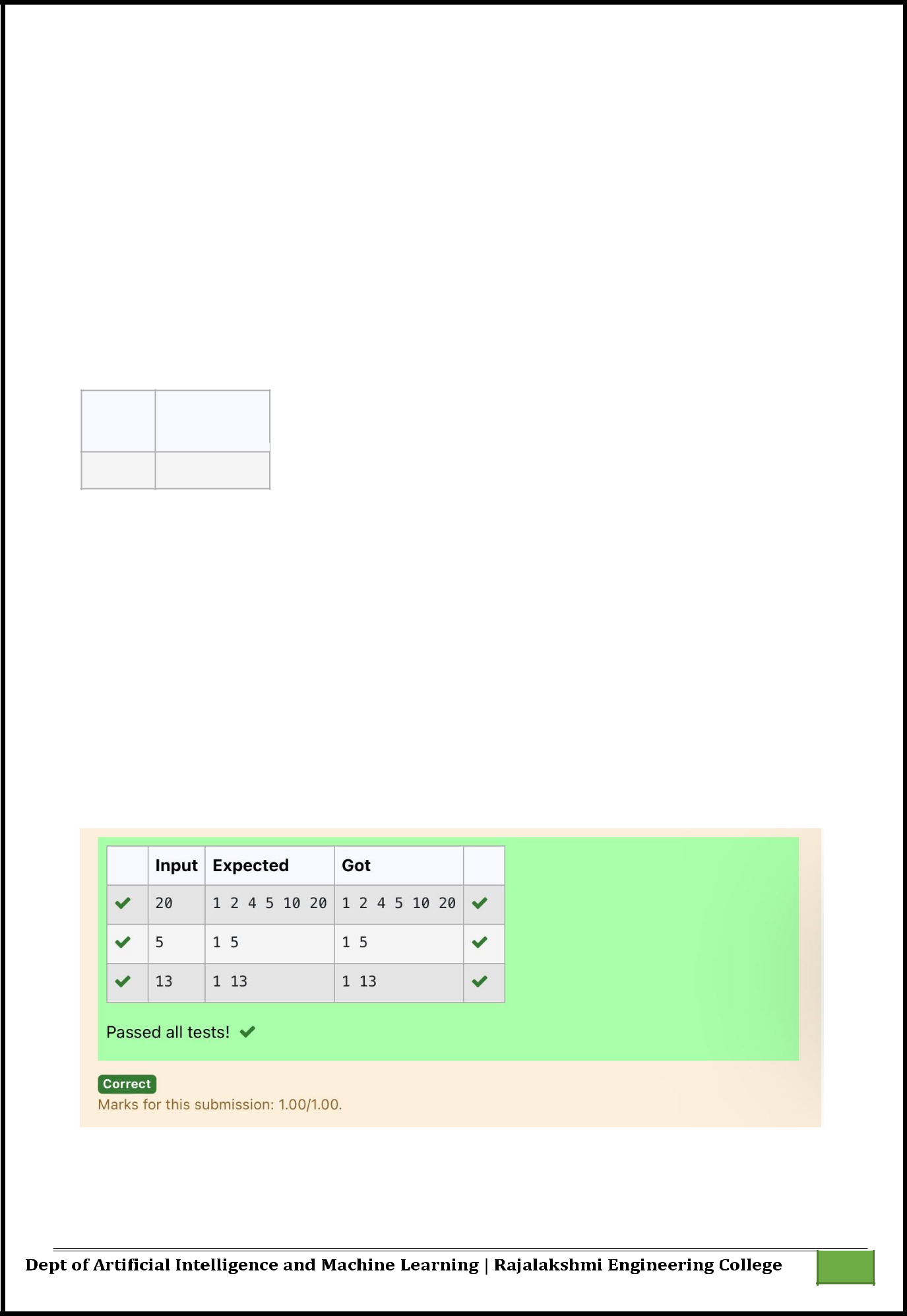


|  |  |
| --- | --- |
| **04** | **- Iteration Control Structures** |
|  | . 60 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.1** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | [**Name: Edmond**](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5720) **Allan A** |
|  |  |  |  |

**Factors of a number**

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number).

**For example:**

**Input Result**

20 12451020 

**PROGRAM**

a=int(input())

for i in range(1,a+1):

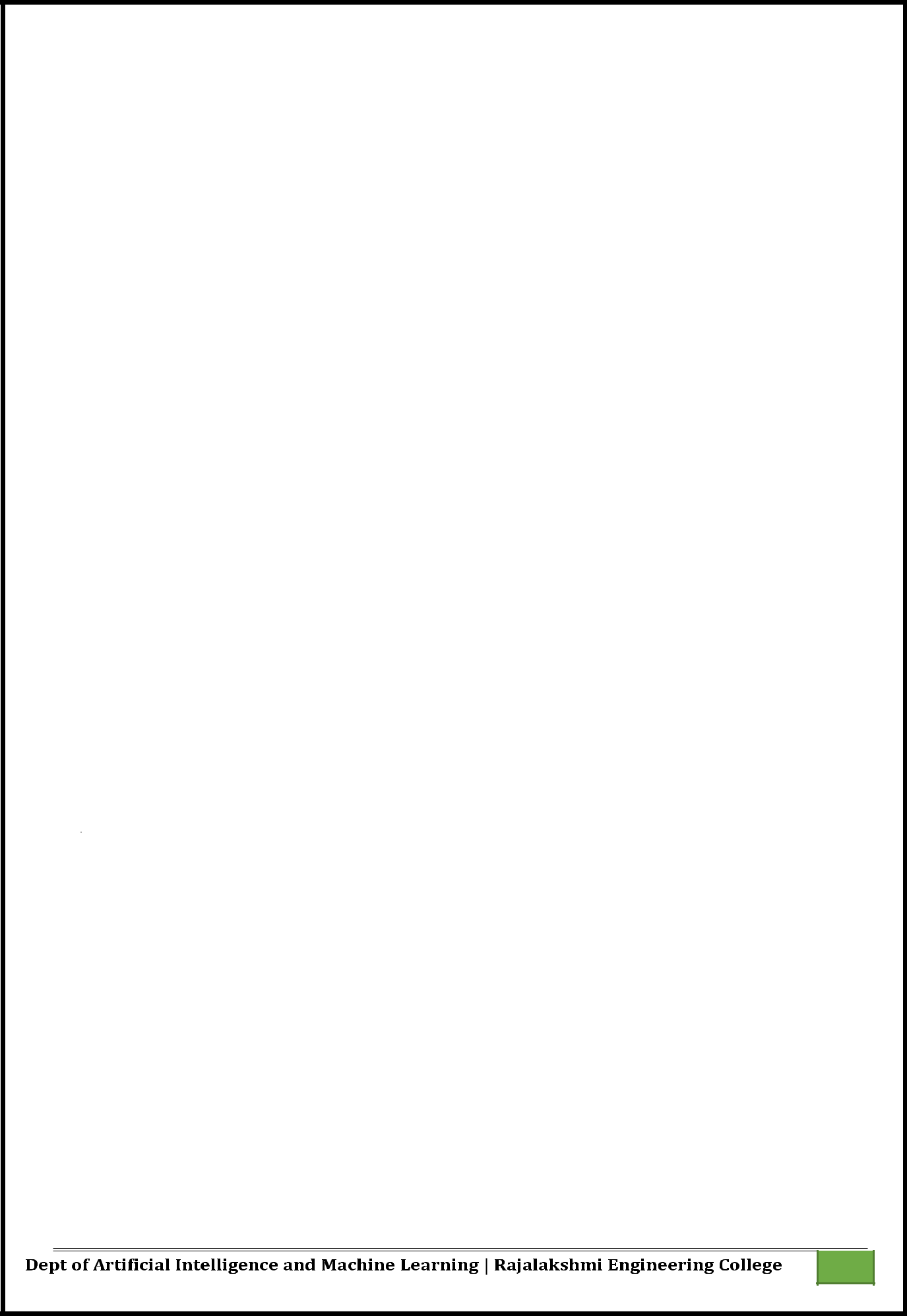
if (a%i==0):

print(i,end=’ ‘)

Output:

. 61

|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.2** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |



**Non Repeated Digit Count**

Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.

Assumption: The input number will be a positive integer number >= 1 and <= 25000.

Some examples are as below.

If the given number is 292, the program should return 1 because there is only 1 non-- repeated digit '9' in this number

If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.

If the given number is 108, the program should return 3 because there are 3 non-- repeated digits in this number, '1', '0', and '8'.

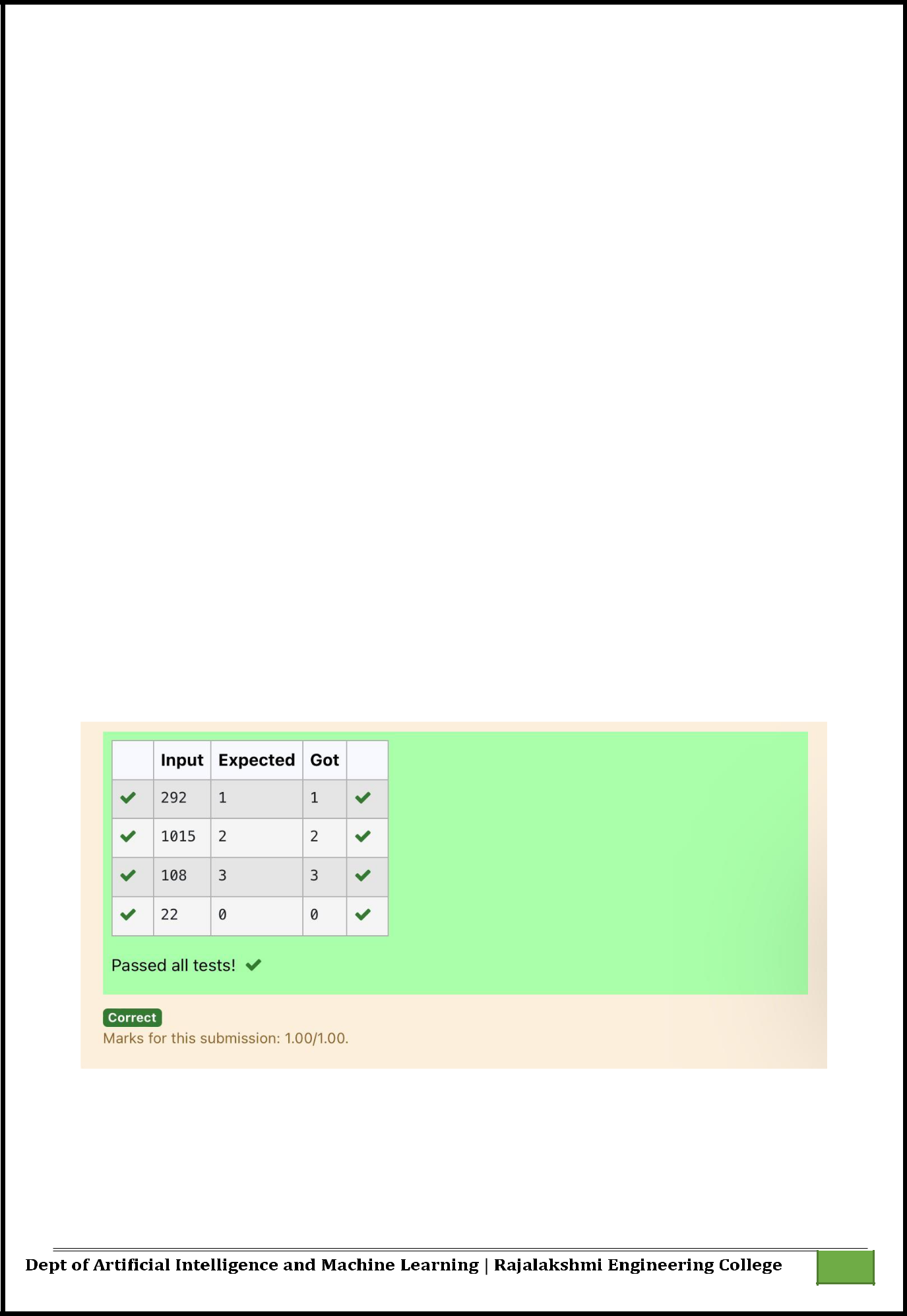
If the given number is 22, the function should return 0 because there are NO non-- repeated digits in this number.

**For example:**

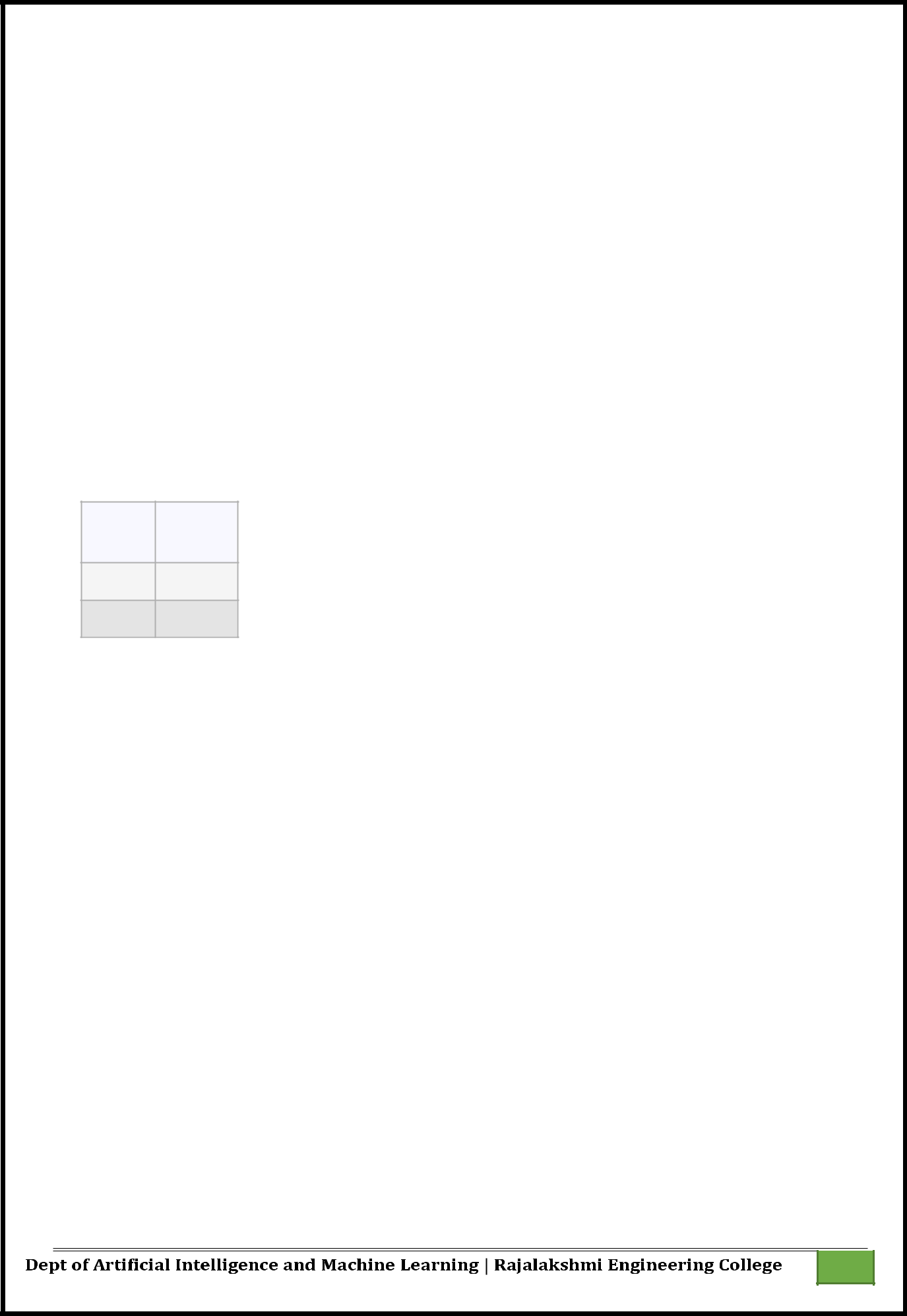
|  |  |
| --- | --- |
| **Input** | **Result** |
|  |  |
| 292 | 1 |
|  |  |
| 1015 | 2 |
|  |  |
| 108 | 3 |
|  |  |
| 22 | 0 |
|  |  |

. 62

|  |  |
| --- | --- |
| **PROGRAM** |  |
| num=int(input()) |  |
| count=0 |  |
| last=len(str(num)) |  |
| for i in range(1,last): |  |
| temp=num%10 |  |
| num=num//10 |  |
| if (str(temp)not in str(num)): |  |
| count+=1 |  |
| if(len(str(num))==1 and count==last-1): |  |
| print(count+1) |  |
| else: |  |
| print(count) |  |
| Output: |  |
| . | 63 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.3** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |



**Prime Checking**

Write a program that finds whether the given number N is Prime or not. If the number is prime, the program should return 2 else it must return 1.

Assumption: 2 <= N <=5000, where N is the given number.

Example1: if the given number N is 7, the method must return 2

Example2: if the given number N is 10, the method must return 1

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| 7 | 2 |
| 10 | 1 |

**PROGRAM**

a=int(input())

count=0

for i in range(2,a):

if (a%i==0):

count+=1

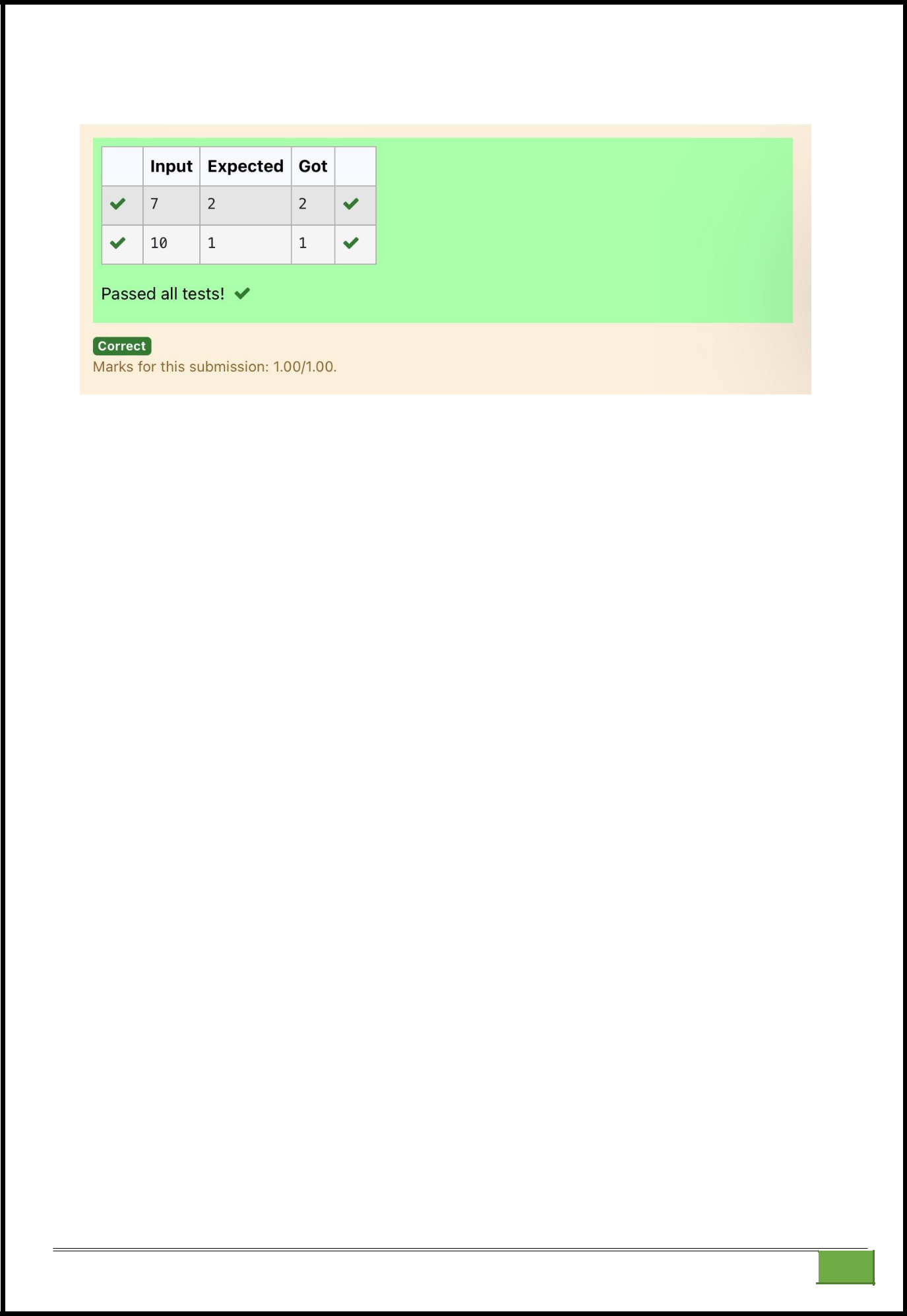
if(count==0):

print(“2”)

else:

print(“1”)

. 64

**Output:**

. 65



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.4** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |

**Next Perfect Square**

Given a number N, find the next perfect square greater than N.

Input Format:

Integer input from stdin.

Output Format:

Perfect square greater than N.

Example Input:

10

Output:

16

**PROGRAM**

from math import sqrt

num=int(input())

while True:

a=int(sqrt(num))

if(num==pow(a,2)):

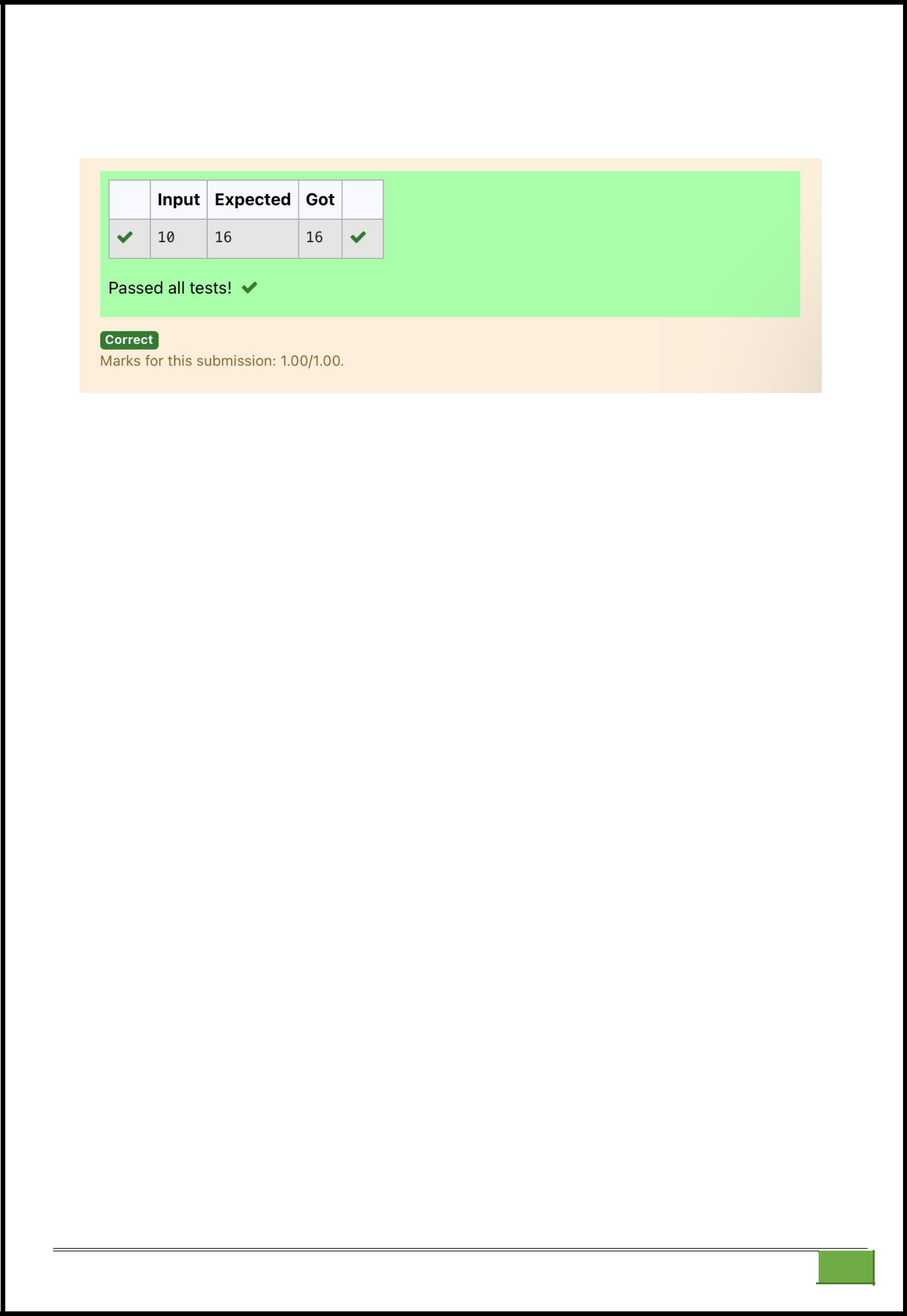
print(num)

break

else:

num+=1

. 66

**Output:**

. 67

|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.5** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |



**Nth Fibonacci**

Write a program to return the nth number in the fibonacci series. The value of N will be passed to the program as input.

NOTE: Fibonacci series looks like –

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . and so on.

i.e. Fibonacci series starts with 0 and 1, and continues generating the next number as the sum of the previous two numbers.

* first Fibonacci number is 0,
* second Fibonacci number is 1,
* third Fibonacci number is 1,
* fourth Fibonacci number is 2,
* fifth Fibonacci number is 3,
* sixth Fibonacci number is 5,
* seventh Fibonacci number is 8, and so on.

**For example:** Input Output

78

**PROGRAM**

a=int(input())

b=0

c=1

d=0

for i in range(3,a+1):

d=c+b

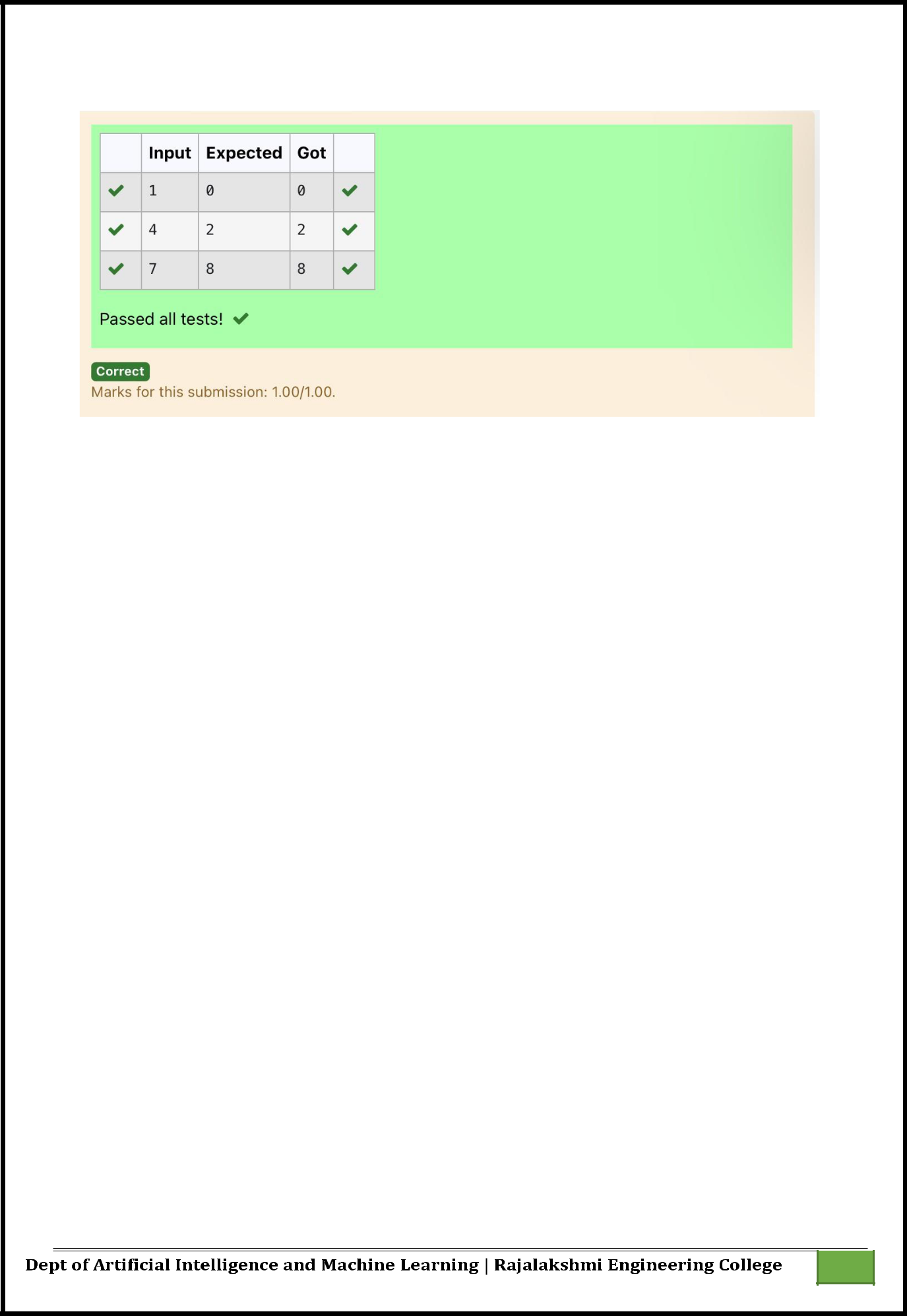
b=c

c=d

print(d)

. 68

|  |  |
| --- | --- |
| **Output:** |  |
| . | 69 |





|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.6** | **Date:** |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |

**Disarium Number**

A Number is said to be Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. Write a program to print number is Disarium or not.

Input Format:

Single Integer Input from stdin.

Output Format:

Yes or No.

Example Input:

175

Output:

Yes

Explanation

1^1 + 7^2 +5^3 = 175

Example Input:

123

Output:

No

For Example:

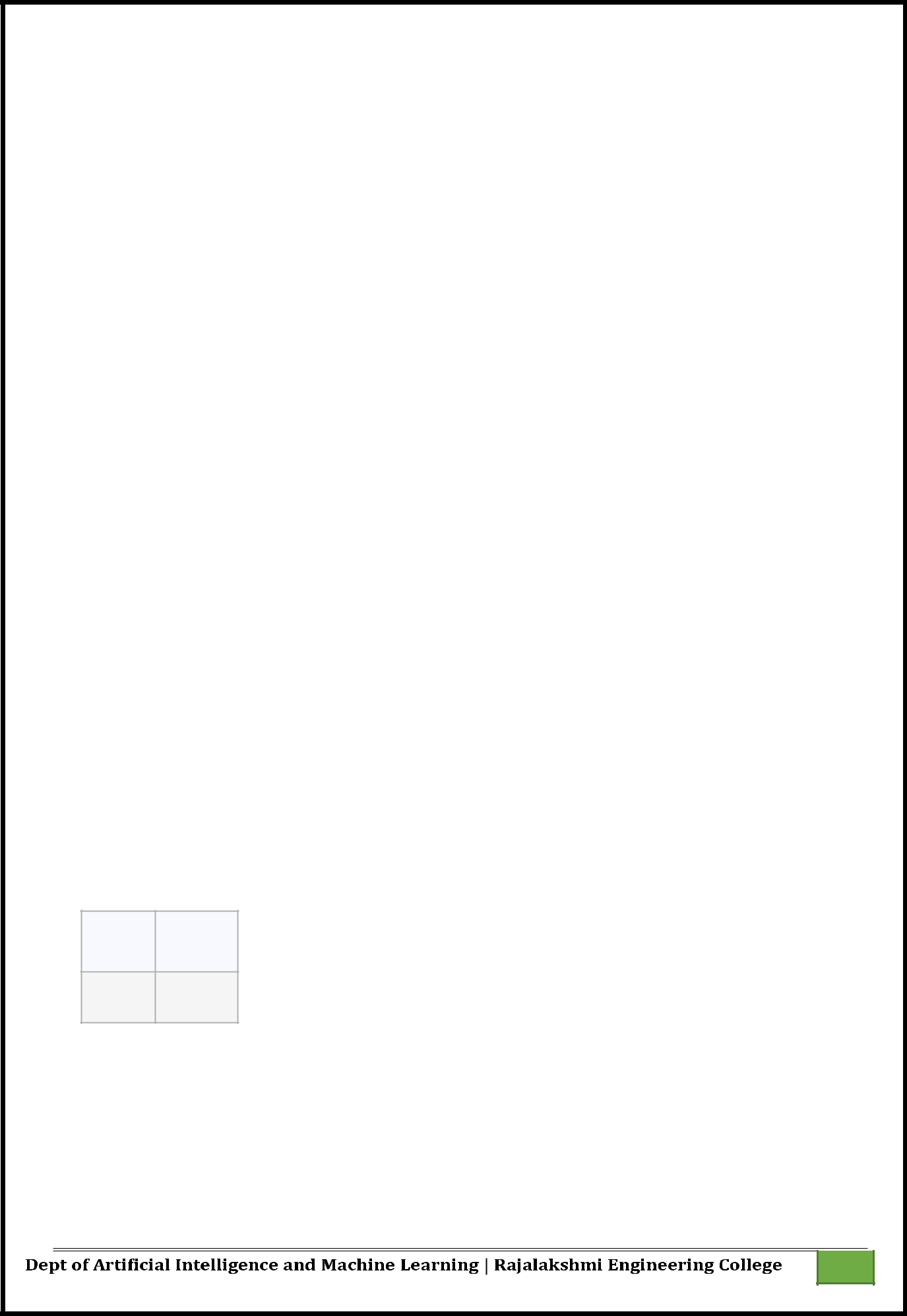
Input Result

1. Yes
2. No

. 70



|  |  |
| --- | --- |
| **PROGRAM** |  |
| num=int(input()) |  |
| last=len(str(num)) |  |
| temp=num |  |
| Sum=0 |  |
| for i in range(0,last): |  |
| n=temp%10 |  |
| temp=temp//10 |  |
| sum=sum+(pow(n,last-i)) |  |
| if (sum==num): |  |
| print(“Yes”) |  |
| else: |  |
| print(“No”) |  |
| **Output:** |  |
| . | 71 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.7** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |

**Sum of Series**

Write a program to find the sum of the series 1 +11 + 111 + 1111 + . . . + n terms (n will be given as input from the user and sum will be the output)

Sample Test Cases

Test Case 1

Input

4

Output

1234

Explanation:

As input is 4, have to take 4 terms.

1+11+111+1111

Test Case 2

Input

6

Output

123456

**For example:**

**Input Result**

3

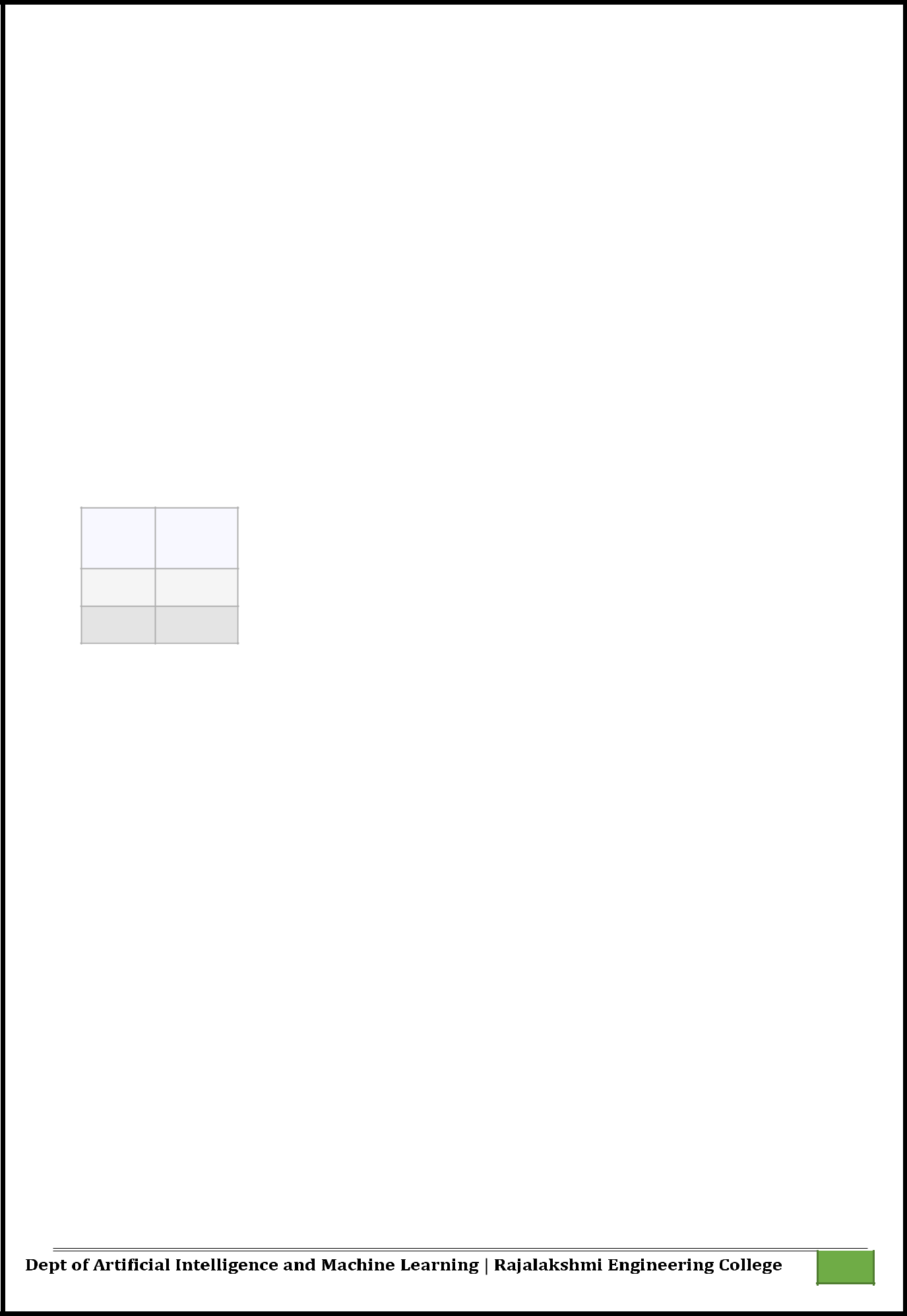
123

. 72

|  |  |
| --- | --- |
| **PROGRAM** |  |
| num=int(input()) |  |
| sum1=0 |  |
| for i in range(1,num+1): |  |
| st=’1’\*i |  |
| sum1=sum1+int(st) |  |
| print(sum1) |  |
| Output: |  |
| . | 73 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.8** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |



**Unique Digit Count**

Write a program to find the count of unique digits in a given number N. The number will be passed to the program as an input of type int.

Assumption: The input number will be a positive integer number >= 1 and <= 25000.

For e.g.

If the given number is 292, the program should return 2 because there are only 2 unique digits '2' and '9' in this number

If the given number is 1015, the program should return 3 because there are 3 unique digits in this number, '1', '0', and '5'.

**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| 292 | 2 |
| 1015 | 3 |

**PROGRAM**

num=int(input())

count=0

last=len(str(num))

for i in range(1,last):

n=num%10

num=num//10

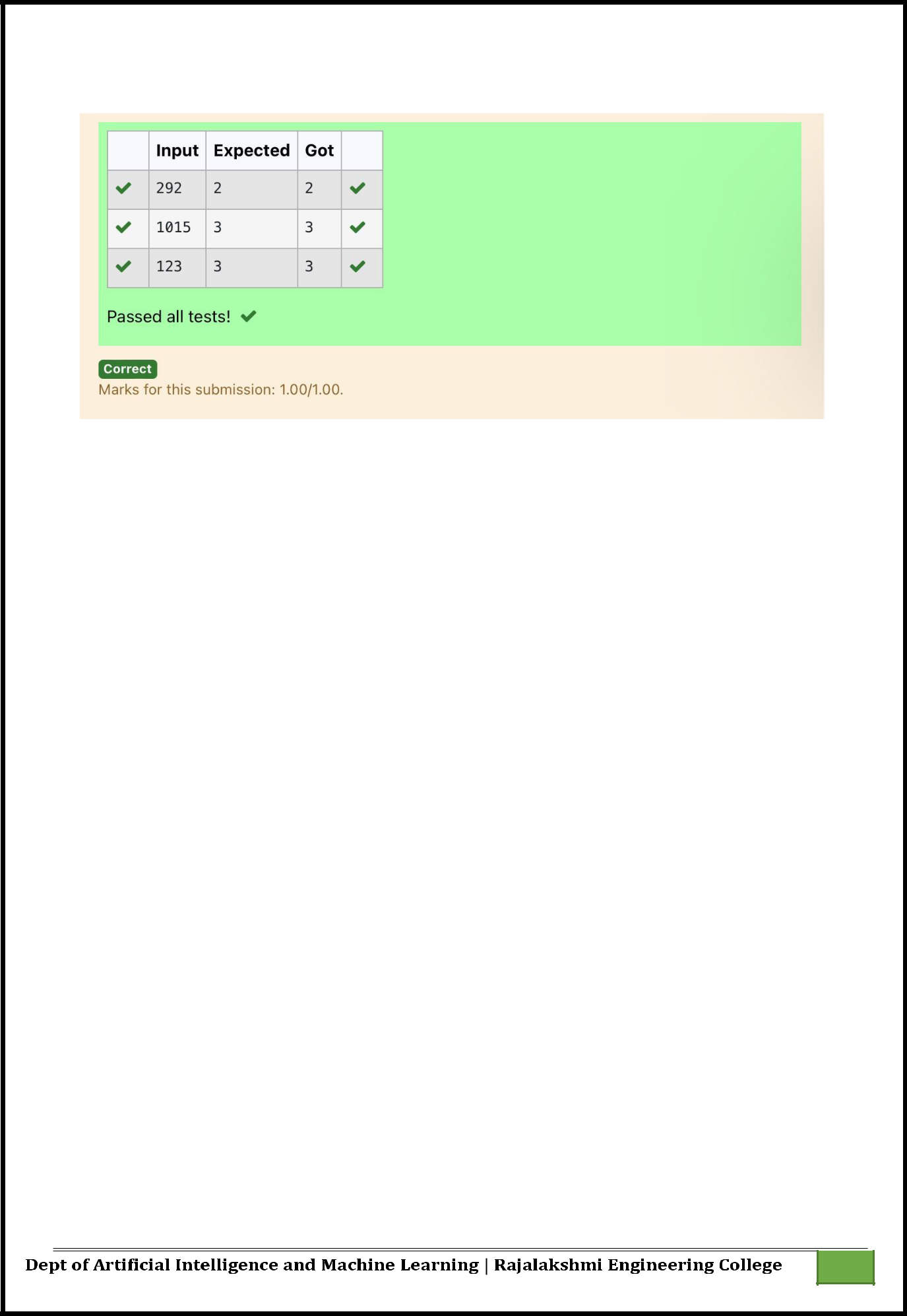
if (str(n)not in str(num)):

count+=1

print(count+1)

. 74

|  |  |
| --- | --- |
| **Output:** |  |
| . | 75 |





|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.9** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |

**Product of single digit**

Given a positive integer N, check whether it can be represented as a product of single digit numbers.

Input Format:

Single Integer input.

Output Format:

Output displays Yes if condition satisfies else prints No.

For Example:

Input Output

1. Yes
2. No

**PROGRAM**

a=int(input())

count=0

for i in range(1,10):

for j in range(1,10):

if (i\*j==a):

print(“Yes”)

count+=1

break

if(count>0):

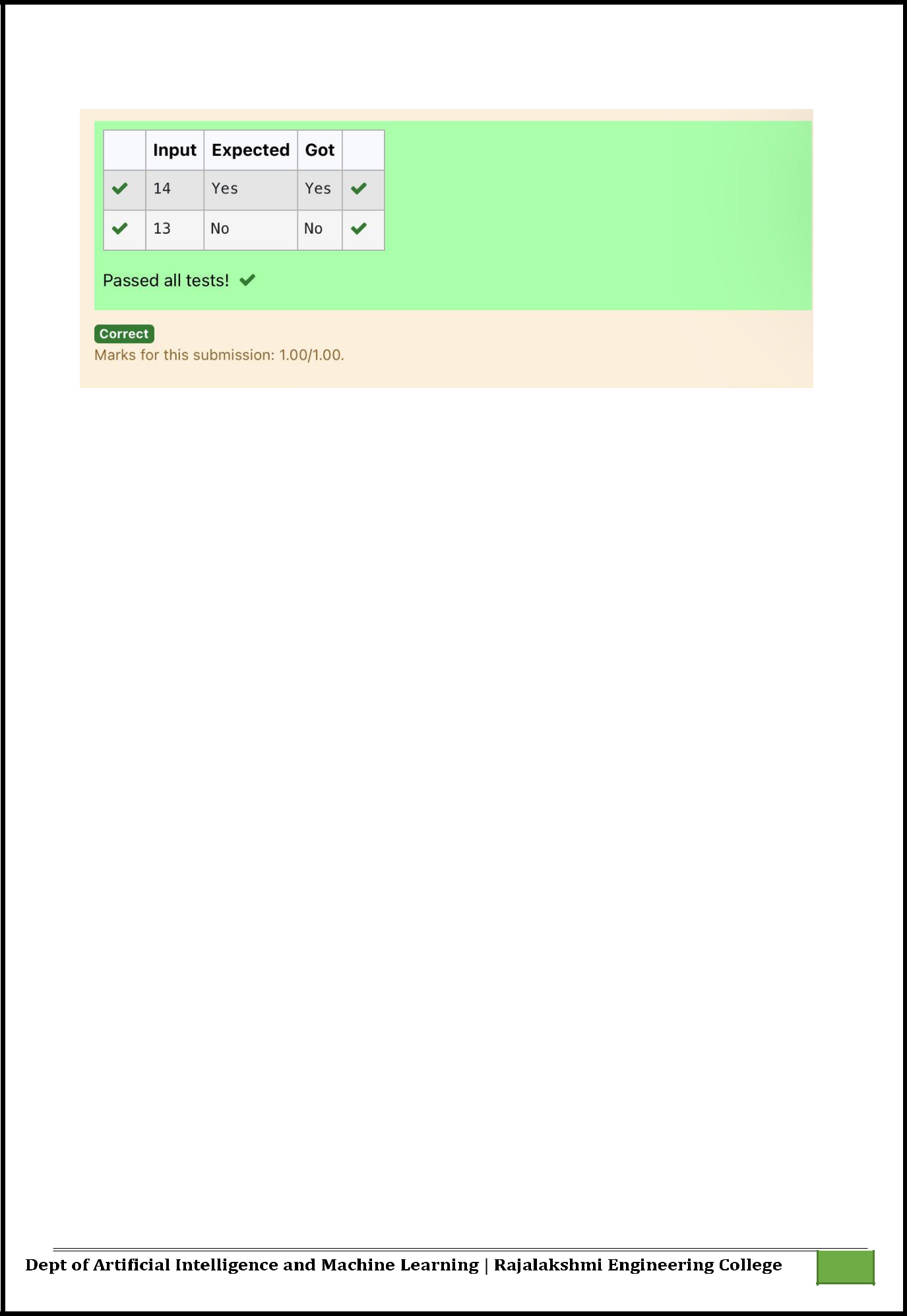
break

if(count==0):

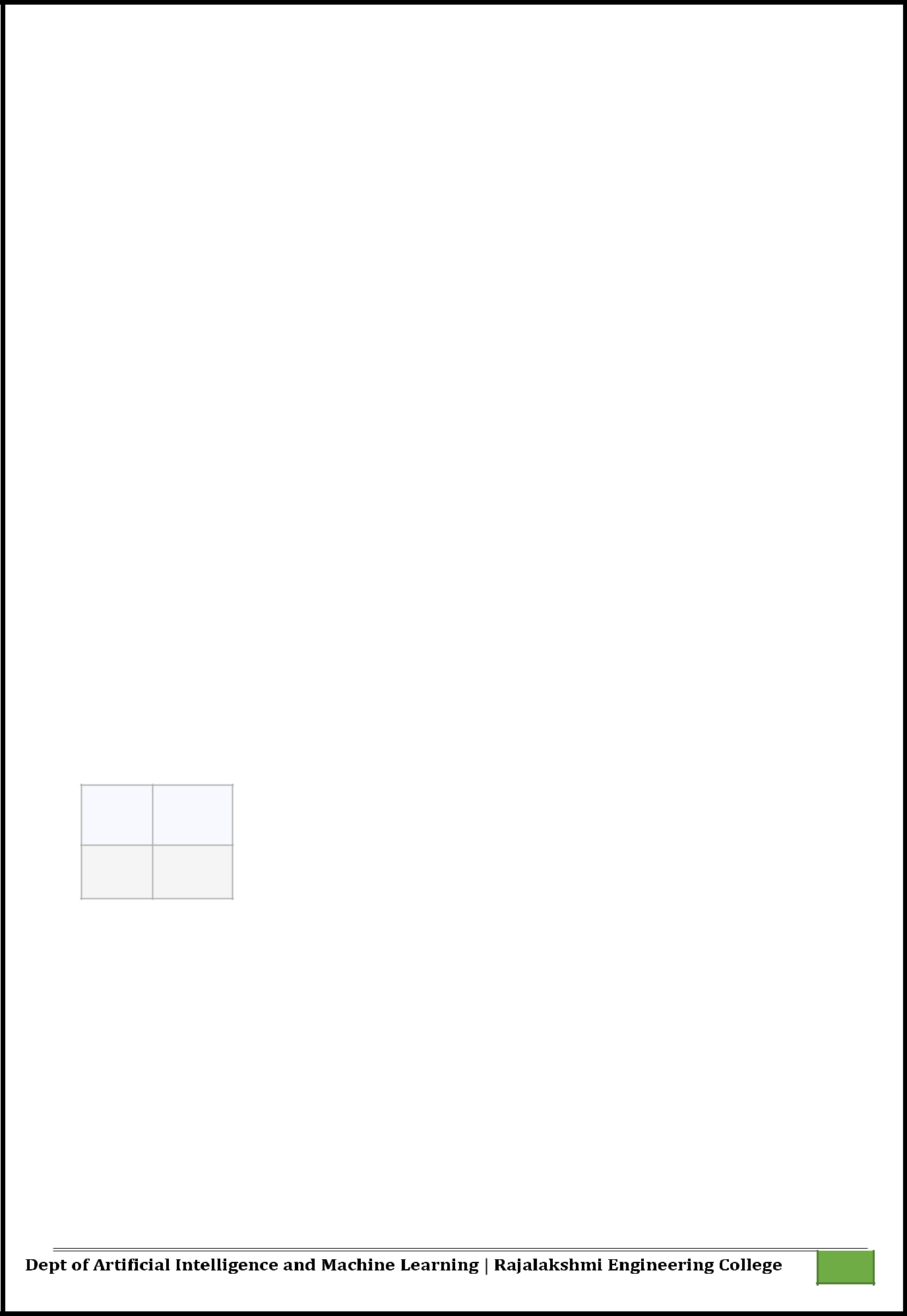
print(“No”)

. 76

|  |  |
| --- | --- |
| Output: |  |
| . | 77 |



|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No.** | **:** | **4.10** | **Date:** 30/3/24 |
| **Register No.:231501044** | | | **Name: Edmond Allan A** |
|  |  |  |  |



**Perfect Square After adding One**

Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

Input Format:

Single integer input.

Output Format:

Yes or No.

Example Input:

24

Output:

Yes

Example Input:

26

Output:

No

**For example:**

**Input Result**

24

Yes

**PROGRAM**

from math import sqrt

num=int(input())

fin=num+1

sq=int(sqrt(fin))

if (fin==pow(sq,2)):

print(“Yes”)

else:

print(“No”)

. 78

|  |  |
| --- | --- |
| Output: |  |
| . | 79 |

