

# Worksheet: Iteration

CS 101 Algorithmic Problem Solving

Fall 2023

Name(s): \_\_\_\_\_

HU ID (e.g., xy01042): \_\_\_\_\_

## 1. What is Ali doing

Ali picks out a random number to see whether it is divisible by 1 and itself or not, Ali is facing some difficulty so it's your job to help him out.

Given a number  $N$  which Ali picks out help Ali find out if it is divisible by 1 and itself or not, if what Ali wants is True return True else return False.

### Constraints

- $1 \leq N \leq 10000$

### Interaction

The input comprises a single line containing a number of  $N$

The output should contain true or false depending on whether the number is divisible by 1 and itself.

### Sample

Input	Output
3	True
8	False

In the first case,  $N = 3$ , 3 is only divisible by 1 and itself so True is the output

In the second case,  $N = 8$ , 8 is divisible by 2 and 4, meaning that it does not satisfy what Ali wants so False.

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
36	False
51	False
107	True

### Pseudocode

```
1     flag = False
2
3     if N == 1:
4         return "False"
```

```

5
6     elif N > 1:
7         for i in range(2,N):
8             if (N % i) ==0:
9                 flag = True
10                break
11        if flag:
12            return "False"
13
14    else:
15        return "True"

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your psuedocode in the space below.

```

1    N = 36
2    flag = False
3
4    if N ==1:
5        return "False"
6
7    elif 36 > 1: #True
8        for i in range(2,36): #i= 2
9            if (36 % 2) ==0: #True
10               flag = True
11               break
12        if flag: #True
13            return "False" #Expected output!
14        else:
15            return "True"

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:** Given a number N, we want to find out if it's a prime number. We check if it can be divided by any number other than 1 and itself. If it can be divided by such a number, we say it's not prime (return False); otherwise, we say it's prime (return True).

## 2. Number lover

Ali is a number lover so much that he wants to reverse a given number.

Although he loves numbers he is having a hard time doing this using iteration/repetition, Your job is to figure out how to do so and help Ali reverse his favourite numbers.

### Constraints

- $1 \leq N \leq 10^5$

### Interaction

The input contains a single integer value denoting Ali's favourite number

Output should contain the reversed number.

### Sample

Input	Output
12345	54321
33602	20633

In the first case,  $N = 12345$ , reversing it would make 54321

In the second case,  $N = 33602$ , reversing it would make 20633

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
8273	3728
91872	27819

### Pseudocode

```

1  reversed = 0
2  while N > 0:
3      remainder = N % 10
4      reversed = (reversed*10) + remainder
5      N = N // 10
6  return reversed

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your psuedocode in the space below.

```

1  N = 8273
2  reversed = 0
3  while 8273 > 0: #True
4      remainder = 8273 % 10 # remainder = 3, 7, 2, 8
5      % reversed = (reversed*10) + remainder # reversed = 3, 37, 372, 3728
6      % N = N // 10 # N = 827, 82, 8
7  return reversed # 3728, which is the expected output

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:** Given input  $N$ , we have to reverse it by iteratively extracting its last digit (remainder) using the modulus operator (%), and then adding that digit to the 'reversed' variable after shifting the existing reversed number left by one position (multiplying it by 10). This process continues until  $N$  becomes zero, resulting in the reversed number.

## 3. Armstrong numbers!

A friend of yours is interested in Armstrong numbers. Armstrong numbers are numbers that if the sum of cubes of each digit of the number is equal to the number itself.

Your friend wants to investigate and figure out how many Armstrong numbers are between 1 and  $N$ .

### Constraints

- $1 \leq N \leq 999$

### Interaction

The input contains a single digit which is  $N$ .

Output should contain multiple lines if there are multiple Armstrong numbers from smallest to largest on each line.

### Sample

Input	Output
1	1
100	1

In the first case,  $N = 1$ , 1 is an Armstrong number as  $1^3 = 1$

In the second case,  $N = 100$ , The only Armstrong number between 1 to 100 is 1 itself with the reasoning as part 1.

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
Input	Output
200	1, 153
101	1
219	1, 153

### Pseudocode

```

1  for number in range(1,N+1):
2      temp = number
3      digit1 = temp%10
4      temp = temp//10
5      digit2 = temp%10
6      temp = temp//10
7      digit3 = tem%10
8
9      if ((digit1**3) + (digit2**3) + (digit3**3) == number):
10         return number

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your psuedocode in the space below.

```

1  N = 101
2  for number in range(1,200+1): # 1- 101
3      temp = number # 1
4      digit1 = temp%10 # 1
5      temp = temp//10 # 0
6      digit2 = temp%10 # 0
7      temp = temp//10 # 0
8      digit3 = tem%10 # 0
9
10     if ((1**3) + (0**3) + (0**3) == number): # ( 1 + 0 + 0 ) = 1
11         return number # 1 which is expected output!

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:** Given an input  $N$ , we need to find all numbers between 1 and  $N$  (inclusive) that are equal to the sum of the cubes of their individual digits. We do so by extracting and evaluating the digits of each number and comparing the sum of their cubes to the original number.

## 4. Working your way up

In your math class today you were asked to calculate the sum of digits of a particular number,

your job is now to write psuedocode to do that for you so that you can easily complete your math homework assignment.

### Constraints

- $1 \leq N \leq 10^8$

### Interaction

The input contains a single integer value denoting the number whose digits you have to sum

Output would contain the sum of the digits.

### Sample

Input	Output
1829	20
23	5

In the first case,  $N = 1829$ , therefore the answer would be  $1 + 8 + 2 + 9 = 20$

In the second case,  $N = 23$ , therefore the answer would be  $2 + 3 = 5$

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
2872	19
98723	29

### Pseudocode

```

1  sum = 0
2  while N > 0:
3      digit = N % 10
4      sum += digit
5      N = N // 10
6  return sum

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```

1  N = 2872
2  sum = 0
3  while 2872 > 0: # While True
4      digit = N % 10 # 2, 7, 8, 2
5      sum += digit # 2, 9, 17, 19
6      N = N // 10 # 287, 28, 2
7  return sum # 19, which is the expected output

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

Given an input  $N$ , we have to compute the sum of its digits by repeatedly extracting and adding the digits of the number until all digits have been processed, resulting in the final sum, which is then returned.

## 5. Digit counter!

Ever found yourself curious about the number of digits hiding within a mysterious number? Instead of painstakingly counting them one by one, you're about to embark on a fascinating journey to create an algorithm that will seamlessly unveil the secret, making it a breeze to determine the number of digits in any given number.

### Constraints

- $1 \leq N \leq 10^8$

### Interaction

The input contains a single integer value denoting the number whose digits are to be counted.

Output would contain the number of digits present in the integer.

### Sample

Input	Output
19	2
23871	5

In the first case,  $N = 19$ , there are two digits meaning the answer is 2.

In the second case,  $N = 23871$ , there are five digits meaning the answer is 5.

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
28728172	8
4827123421	10

### Pseudocode

```

1  digits = 0
2  while N > 0:
3      digits += 1
4      N = N // 10
5  return digits

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```

1  N = 28728172
2  digits = 0
3  while 28728172 > 0: # While True
4      digits += 1 # 1,2,3,4,5,6,7,8
5      N = N // 10 # 2872817, 287281, 28728, 2872, 287, 28, 2
6  return digits # 8, which is the expected output

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:** Given a number  $N$ , we need to output the total digits in that number by slicing and summing up each digit encountered.

## 6. Lucas Series

Your friend introduces you to the fascinating Lucas Series, a sequence that commences with the numbers 2 and 1, where each subsequent term is calculated as the sum of the last two terms. Intrigued, your friend challenges you to create an algorithm capable of revealing the  $n$ th term of the Lucas Series. With boundless enthusiasm, you confidently declare that it's not only possible but also your mission. Your task, given  $N$ , is to generate and output the  $n$ th term of this captivating Lucas series.

### Constraints

- $1 \leq N \leq 10^3$

### Interaction

The input contains a single integer value denoting the value of  $N$

Output would contain the  $n$ th term of the lucas series.

### Sample

Input	Output
3	3
6	11

In the first case,  $N = 3$ , starting from 2, 1, the third term would be  $2 + 1 = 3$

In the second case,  $N = 6$ , starting from 2, 1, the third term would be 3, fourth term would be 4, the fifth would be 7 and the sixth would be 11.

### Exercise

In the space provided, indicate the outputs for the given inputs.

Input	Output
9	76
17	3571

### Pseudocode

```

1  sum = 0
2  if N == 0:
3      return 2
4
5  elif N == 1:
6      return 1
7
8  else:
9      a = 2
10     b = 1
11     for i in range(3, N+2):
12         sum = a+b
13         a = b
14         b = sum
15     return sum

```

### Dry Run

Using any two of the inputs provided in the Exercise section above, dry run your pseudocode in the space below.

```

1  sum = 0
2  N = 9

```

```

3      if N == 0: # False
4          return 2
5
6      elif N == 1: # False
7          return 1
8
9      else:
10         a = 2
11         b = 1
12         for i in range(3, N+2): # 3 - 10
13             sum = a+b # 3, 4, 7, 11, 18, 29, 47, 76
14             a = b # 2, 3, 4, 7, 11, 11, 29
15             b = sum # 3, 4, 7, 11, 18, 29, 47
16         return sum # 76, which is the expected output

```

### Problem Identification

Briefly explain the underlying problem you identified in the above question that led you to your solution.

**Answer:** Given an input N, we need to iteratively sum the last two consecutive numbers to find the value of the nth term in a Lucas series.

## 7. Iterative exponent

You have studied exponents both early in this class and in secondary school, and your friends challenge you to find the exponent using multiplications only. Given m and n two integers your job is to find the exponent is the form  $m^n$

### Constraints

- $1 \leq m, n \leq 10^3$

### Interaction

The input contains two space-separated integers m and n.

Output would contain a single integer which is  $m^n$

### Sample

Input	Output
3,4	81
0,2	0

In the first case,  $m, n = 3, 4$ ,  $3^4 = 81 = 3 * 3 * 3 * 3$

In the second case,  $m, n = 0, 2$ ,  $0^2 = 0 = 0 * 0$

### Proposed Solution

```

1      product = m
2      if m == 0:
3          return 0
4
5      elif n == 0:
6          return 0
7
8      else:
9          for i in range(0, n-1):
10             product = product**m

```



```

11
12         return product

```

### Dry Run

Using the inputs provided in the Sample section above, dry run the proposed code solution below.

```

1  m = 3
2  n = 4
3  product = 3
4  if 3 == 0: # False
5      return 0
6
7  elif 4 == 0: #False
8      return 0
9
10 else:
11     for i in range(0,4-1): #
12         product = product**3 # product = 27, 19683, 7625597484987
13
14     return product # 7625597484987 - not the expected output

```

### Error Identification

Briefly explain the errors you identified in the proposed code solution. Mention the line number and the errors in each line.

Line 10: Symbol (\*\*), which is used for exponentiation or raising numbers to power, is used instead of the multiplication symbol (\*), which should be used for multiplication.

Line 6: To handle the case where the exponent is 0, the function should return 1, as any non-zero number raised to the power of 0 is defined to be equal to 1.

### Correct Solution

Rewrite the lines of code you mentioned above with their errors corrected.

```

1  product = m
2  if m == 0:
3      return 0
4
5  elif n == 0:
6      return 1
7
8  else:
9      for i in range(0,n-1):
10         product = product*m
11     return product

```

## 8. Never Lucky

Maheen loves lucky numbers. Everyone knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits 4 and 7, for example, 47, 744, and 4. Help Maheen find out if a given number is lucky.

### Constraints

- $1 \leq n \leq 10^3$

### Interaction

The input contains one integer the number n.

Output would contain True if the number is lucky and false if it is not. **Sample**

Input	Output
47	True
78	False

In the first case,  $n = 47$ , 47 contains only 4s and 7s.

In the second case,  $n = 78$ , 78 contains the digit 7 but also the digit 8.

### Proposed Solution

```

1  num1 = n
2  count_digits = 0
3  count_true = 0
4
5  while num1!=0:
6      rem = num1%10
7      if rem == 4 or rem == 7:
8          count_true +=1
9          count_digits += 1
10     num1 = num1//10
11
12     if count_true == count_digits:
13         return "True"
14
15     return "False"
```

### Dry Run

Using the inputs provided in the Sample section above, dry run the proposed code solution below.

```

1  num1 = n # 47
2  count_digits = 0
3  count_true = 0
4
5  while 47 != 0: #True
6      rem = num1%10 #7
7      if rem == 4 or rem == 7: # True
8          count_true +=1 # 1
9          count_digits += 1 #1
10     num1 = num1//10 #4
11
12     if count_true == count_digits: # True
13         return "True" # True
14
15     return "False" # True
16         # False (not expected output as we want True or False)
```

### Error Identification


Briefly explain the errors you identified in the proposed code solution. Mention the line number and the errors in each line.

Line 17: The `return 'False'` statement is not indented correctly. It should be part of an `else` block so that it only executes if the condition `count_true == count_digits` is `False`. As it stands, it will always return `'False'` regardless of whether the number is lucky or not.

**Correct Solution**

Rewrite the lines of code you mentioned above with their errors corrected.

```
1 num1 = n
2 count_digits = 0
3 count_true = 0
4
5 while num1 != 0:
6     rem = num1 % 10
7     if rem == 4 or rem == 7:
8         count_true += 1
9     count_digits += 1
10    num1 = num1 // 10
11 if count_true == count_digits:
12     return "True"
13 else:
14     return "False"
```



**Rough Work**

SAMPLE SOLUTION