Q. 64. Write a Python program to find an integer exponent x such that a^x = n.

Input: a = 2 : n = 1024 Output: 10 Input: a = 3 : n = 81 Output: 4

In [11]:
```
a=int(input('Enter number to whose power is needed:'))
n=int(input('Enter the value od that'))
m = 1
x = 0
while m != n:
    x += 1
    m *= a
print(x)
```

```
Enter number to whose power is needed:2
Enter the value od that1024
10
```

Q. 181 Income Tax Calculator

In [12]:
```
#Q.3.
grade_level=input("Enter the grade_level (A,B,C,D,E or F:)")
#Assigning City 1,2 or 3 for House Rent Calculation
print("city 1 is a tier 1 (metro), city 2 is tier 2 and city 3 is tier 3")
city=int(input("Enter the city (1,2 or 3)"))
#Assigning Transport allowance (TA)
TA=900
#Assigning Professional Tax (PT)
PT=200
if grade_level=="A":
    #Assigning basic pay (BP) in Rs. for different grade levels
    BP=60000
    Allowance=8000
elif grade_level=="B":
    BP=50000
    Allowance=7000
elif grade_level=="C":
    BP=40000
    Allowance=6000
elif grade_level=="D":
    BP=30000
    Allowance=5000
elif grade_level=="E":
    BP=20000
    Allowance=4000
elif grade_level=="F":
    BP=10000
    Allowance=3000
else:
    print("Enter correcrt grade_level (A,B,C,D,E,F)")
#Assigning hra allowance
if city==1:
    hra=0.3*BP
```

```
elif city==2:
    hra=0.2*BP
elif city==3:
    hra=0.1*BP
#Assigning Dearness Allowance (DA)
DA=0.5*BP
#Assigning Employees Provident Fund(EPF)
EPF=0.11*BP
gross_pay=BP+hra+DA+Allowance+TA-PT-EPF
print("Gross Pay of an Employee is:",gross_pay)
Annual_pay=gross_pay*12
print("Annual income of an employee is:",Annual_pay)
#Making an Income Tax Calculator
if Annual_pay<=250000:
    print("No income Tax")
elif 250000<Annual_pay<=500000:
    Income_Tax=(Annual_pay-250000)*0.05
elif 500000<Annual_pay<=750000:
    Income_Tax=(Annual_pay-500000)*0.1+12500
elif 750000<Annual_pay<=1000000:
    Income_Tax=(Annual_pay-750000)*0.15+37500
elif 1000000<Annual_pay<=1250000:
    Income_Tax=(Annual_pay-1000000)*0.2+75000
elif 1250000<Annual_pay<=1500000:
    Income_Tax=(Annual_pay-1250000)*0.25+125000
elif Annual_pay>=1500001:
    Income_Tax=(Annual_pay-1500000)*0.3+187500
print("Income Tax to be paid by an employee is:",Income_Tax)
```

```
Enter the grade_level (A,B,C,D,E or F:)A
city 1 is a tier 1 (metro), city 2 is tier 2 and city 3 is tier 3
Enter the city (1,2 or 3)1
Gross Pay of an Employee is: 110100.0
Annual income of an employee is: 1321200.0
Income Tax to be paid by an employee is: 142800.0
```

Q. 182.

Write a python program to print all numbers between 1 and 100 (including 1 and 100) that are both, Disarium and Harshad numbers. A number is said to be a Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. For example, 175 is a Disarium number as follows: 11+ 72 + 53 = 1+ 49 + 125 = 175 A harshad number is a number that is divisible by the sum of its digits. E.g., the number 18 is a harshad number, because the sum of the digits 1 and 8 is 9 (1 + 8 = 9), and 18 is divisible by 9.

In [3]:
```
def calculate_length(num):
    length = 0
    while num > 0:
        length += 1
        num //= 10
    return length

def is_disarium_number(num):
    temp = num
    length = calculate_length(num)
```

```
    total = 0

    while temp > 0:
        digit = temp % 10
        total += digit ** length
        temp //= 10
        length -= 1

    return total == num

def is_harshad_number(num):
    temp = num
    digit_sum = 0

    while temp > 0:
        digit_sum += temp % 10
        temp //= 10

    return num % digit_sum == 0

def find_disarium_harshad_numbers(start, end):
    num = start

    while num <= end:
        if is_disarium_number(num) and is_harshad_number(num):
            print(num, end=", ")
        num += 1

# Find and print Disarium and Harshad numbers between 1 and 100
print("Disarium and Harshad numbers between 1 and 100:", end=" ")
find_disarium_harshad_numbers(1, 100)
```

```
Disarium and Harshad numbers between 1 and 100: 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Q. 183. Ask the user to enter 10 test scores. Write a program to do the following:

a) If user enters score greater than 100, then give warning to user that entered score is more than 100 and take that input again from user. b) Print out the highest and lowest scores. c) Print out the average of the scores. d) Print out the second largest score. e) Drop the two lowest scores and print out the average of the rest of them. Note: Use of Python Data structures like string, list, tuple etc. and their inbuilt function is not allowed. For Ex. If Input is like following: Enter Test Score: 80 Enter Test Score: 65 Enter Test Score: 98 Enter Test Score: 70 Enter Test Score: 93 Enter Test Score: 130 Entered score is more than hundred, so enter again Enter Test Score: 95 Enter Test Score: 50 Enter Test Score: 40 Enter Test Score: 75 Enter Test Score: 72 Output should be: Highest Score is: 98 Lowest Score is: 40 Average Test Score is: 73.8 Second Largest Score is: 95 Average after dropping the two lowest scores: 81.0

In [20]:
```
# Function to check if a character is a digit
def is_digit(char):
    return '0' <= char <= '9'

# Function to convert a string to an integer
```

```
def string_to_int(s):
    result = 0
    for char in s:
        # Convert character to integer without using ord()
        digit_value = 0
        for i in range(10):
            if char == str(i):
                digit_value = i
                break
        result = result * 10 + digit_value
    return result

# Function to get a valid positive integer score from the user
def get_valid_score():
    while True:
        score_str = input("Enter Test Score: ")
        valid = True

        # Check if the string represents a positive integer
        for char in score_str:
            if not is_digit(char):
                valid = False
                break

        # Convert the string to an integer if it is valid
        if valid:
            score = string_to_int(score_str)
            if 0 <= score <= 100:
                return score

        print("Invalid input. Please enter a valid integer between 0 and 100.")

first_highest = -1
second_highest = -1
first_lowest = 101
second_lowest = 101
total_scores = 0

for i in range(10):
    number = float(input(f"Enter number {i + 1}: "))
    total_scores+=number
    # Check for highest numbers
    if number > first_highest:
        second_highest = first_highest
        first_highest = number
    elif number > second_highest:
        second_highest = number

    # Check for lowest numbers
    if number < first_lowest:
        second_lowest = first_lowest
        first_lowest = number
    elif number < second_lowest:
        second_lowest = number

# Print the results
print(f"First highest: {first_highest}")
print(f"Second highest: {second_highest}")
print(f"First lowest: {first_lowest}")
print(f"Second lowest: {second_lowest}")
```

```
average_without_lowest = (total_scores - first_lowest - second_lowest) / 8
print("Average after dropping the two lowest scores:", average_without_lowest)
```

```
Enter number 1: 89
Enter number 2: 91
Enter number 3: 47
Enter number 4: 45
Enter number 5: 46
Enter number 6: 42
Enter number 7: 12
Enter number 8: 13
Enter number 9: 15
Enter number 10: 17
First highest: 91.0
Second highest: 89.0
First lowest: 12.0
Second lowest: 13.0
Average after dropping the two lowest scores: 49.0
```

Q. 184

Write a program to encode a number by changing the digits in the given positive integer by user. The rule for changing the digits in number will be: If the digit in number is between 0 to 8 than replace the number with 1 to 9 respectively. (incrementing each digit by +1). If the digit is 9, then replace it with 0. To encode a number, replace digits in following manner: For example: Input: 31590218 Output: The number after encoding is: 42601329 For example: Input: 9259 Output: The number after encoding is: 360 For example: Input: 65217001 Output: The number after encoding is: 76328112 Original Digit in Number New Digit after Encoding 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 0 Note: Use of Python Data structures like string, list, tuple etc. and their inbuilt function is not allowed.

In [57]:
```
# Input a positive integer from the user
original_number = -1  # Initialize to an invalid value

# Keep prompting until a valid positive integer is entered
while original_number < 0:
    user_input = input("Enter a positive integer:")

    # Check if the input is not empty and consists of digits
    if user_input:
        is_valid_input = True
        for char in user_input:
            if '0' > char or char > '9':
                is_valid_input = False
                break

        if is_valid_input:
            original_number = int(user_input)
            if original_number < 0:
                print("Please enter a positive integer.")
        else:
            print("Invalid input. Please enter a positive integer.")
    else:
        print("Invalid input. Please enter a positive integer.")
```

```
    temp=user_input
    count=0
    c=''
    for i in temp:
        while i=='0':
            i='1'
            if i=='1':
                c=c+i
        #print(i,end="")
    #print(c,end="")
    #print(count)

# Encode the number
encoded_number = 0
multiplier = 1  # Used to build the encoded number

while original_number > 0:
    digit = original_number % 10  # Extract the last digit
    encoded_digit = (digit + 1) % 10  # Apply the encoding rule

    encoded_number += encoded_digit * multiplier

    multiplier *= 10

    original_number //= 10  # Remove the last digit

d=c+str(encoded_number)

# Print encoded numbers

print("The number after encoding is:",int(d))
```

```
Enter a positive integer:000123
The number after encoding is: 111234
```

Q. 185. Write a python program to swap first and last digits of a number using loop.

(for example: input = 123456 then output=623451)

In [2]:
```
def swap_first_last_digits(number):
    # Count the number of digits in the given number
    num_digits = 0
    temp_number = number

    while temp_number > 0:
        temp_number //= 10
        num_digits += 1

    # Swap the first and last digits only if there are more than one digit
    if num_digits > 1:
        # Extract the first digit
        first_digit = number // (10 ** (num_digits - 1))

        # Extract the last digit
        last_digit = number % 10
```

```
        # Calculate the remaining digits between the first and last digits
        middle_digits = (number % (10 ** (num_digits - 1))) // 10

        # Construct the swapped number
        swapped_number = last_digit * (10 ** (num_digits - 1)) + middle_digits * 10 +

        return swapped_number
    else:
        # If there is only one digit, no swapping is needed
        return number

# Example usage
original_number = 123456
result = swap_first_last_digits(original_number)

print(f"Original Number: {original_number}")
print(f"Swapped Number: {result}")
```

```
Original Number: 123456
Swapped Number: 623451
```

Q. 186. Pattern

In [89]:
```
num=int(input('Enter number:'))
for i in range(1,num,1):
    for j in range(0,i,1):
        print('* ',end="")
    for j in range(num,i,-1):
        print('  ',end="")
    for j in range(num,i+1,-1):
        print('  ',end="")
    #for j in range()
    for j in range(0,i,1):
        print('* ',end="")
    print()
print((num*2-1)* "* ")
for i in range(1,num+1,1):
    for j in range(num,i,-1):
        print('* ',end="")
    for j in range(1,i+1,1):
        print('  ',end="")
    for j in range(1,i,1):
        print('  ',end="")
    for j in range(num,i,-1):
        print('* ',end="")
    print()
```

```
Enter number:6
*                     *
* *                 * *
* * *             * * *
* * * *         * * * *
* * * * *     * * * * *
* * * * * * * * * * * *
* * * * *     * * * * *
* * * *         * * * *
* * *             * * *
* *                 * *
*                     *
```

Q. 187 Write a program to implement the calculator for the date of Easter.

The following algorithm computes the date for Easter Sunday for any year between 1900 to 2099. Ask the user to enter a year. Compute the following:

1. a = year % 19
2. b = year % 4
3. c = year % 7
4. d = (19 * a + 24) % 30
5. e = (2 *b* + *4* c + 6 * d + 5) % 7
6. dateofeaster = 22 + d + e Special note: The algorithm can give a date in April. You will know that the date is in April if the calculation gives you an answer greater than 31. (You'll need to adjust) Also, if the year is one of four special years (1954, 1981, 2049, or 2076) then subtract 7 from the date. Eg: Input: Year : 2022 Expected Outcome: 2022-04-17 (i.e. 17th April 2022)

In [40]:
```
# Input a year from the user
year_input = input("Enter a year (between 1900 and 2099): ")

# Check if the input is a valid integer
if year_input:
    is_digit = True
    for char in year_input:
        if '0' <= char <= '9':
            continue
        else:
            is_digit = False
            break

    if is_digit:
        year = int(year_input)

        if 1900 <= year <= 2099:
            # Compute the date of Easter
            a = year % 19
            b = year % 4
            c = year % 7
```

```
            d = (19 * a + 24) % 30
            e = (2 * b + 4 * c + 6 * d + 5) % 7

            date_of_easter = 22 + d + e

            # Adjust if the date is in April
            if date_of_easter > 31:
                date_of_easter -= 31

            # Adjust for special years
            if year == 1954 or year == 1981 or year == 2049 or year == 2076:
                date_of_easter -= 7

            # Print the result
            print(f"The date of Easter Sunday in {year} is: {year}-04-{date_of_easter}")
        else:
            print("Invalid input. Please enter a year between 1900 and 2099.")
    else:
        print("Invalid input. Please enter a valid year.")
else:
    print("Invalid input. Please enter a valid year.")
```

```
Enter a year (between 1900 and 2099): 2022
The date of Easter Sunday in 2022 is: 2022-04-17
```

Q.188 Write a Python program to compute the greatest common divisor (GCD) of two positive integers.

The greatest common divisor (GCD) of two nonzero integers a and b is the greatest positive integer d such that d is a divisor of both a and b; that is, there are integers e and f such that a = de and b = df, and d is the largest such integer. The GCD of a and b is generally denoted gcd(a, b). For example, the greatest common factor of 15 and 10 is 5, since both the numbers can be divided by 5.

In [41]:
```
# Input two positive integers from the user
num1_input = input("Enter the first positive integer: ")
num2_input = input("Enter the second positive integer: ")

# Check if the inputs are valid positive integers
if num1_input and num2_input:
    is_num1_valid = True
    is_num2_valid = True

    for char in num1_input:
        if not ('0' <= char <= '9' or char == '-'):
            is_num1_valid = False
            break

    for char in num2_input:
        if not ('0' <= char <= '9' or char == '-'):
            is_num2_valid = False
            break
```

```
    if is_num1_valid and is_num2_valid:
        num1 = int(num1_input)
        num2 = int(num2_input)

        # Ensure num1 is greater than or equal to num2
        if num1 < num2:
            num1, num2 = num2, num1

        # Compute the GCD using the Euclidean algorithm
        while num2 != 0:
            num1, num2 = num2, num1 % num2

        # Print the GCD
        print(f"The greatest common divisor (GCD) of {num1} and {num2} is: {num1}")
    else:
        print("Invalid input. Please enter valid positive integers.")
else:
    print("Invalid input. Please enter valid positive integers.")
```

```
Enter the first positive integer: 15
Enter the second positive integer: 10
The greatest common divisor (GCD) of 5 and 0 is: 5
```

In [ ]:
```
# Q. 189 Write a python program that prompts the user to enter numbers and stops only
sum and average of the numbers, minimum and maximum number from given numbers entered
Note: you are not allowed to use any built in structures like, list ,tuple etc. or any
For Example: Input: 4,1,5,”QUIT”
Output:
Sum=10
Average=3.333
Minimum number=1
Maximum number=5
```

In [47]:
```
# Initialize variables
count = 0
total = 0
min_num = None
max_num = None

# Prompt the user to enter numbers
while True:
    user_input = input("Enter a number (or 'QUIT' to finish): ")

    # Check if the user wants to quit
    if user_input == "QUIT" or user_input == "quit" or user_input == "Quit":
        break

    # Check if the input is a valid number
    is_valid_number = True
    is_negative = False
    is_decimal = False

    for char in user_input:
        if char == '-':
            if not is_negative:
                is_negative = True
            else:
                is_valid_number = False
                break
        elif char == '.':
```

```
            if not is_decimal:
                is_decimal = True
            else:
                is_valid_number = False
                break
        elif not ('0' <= char <= '9'):
            is_valid_number = False
            break

    if not is_valid_number:
        print("Invalid input. Please enter a valid number.")
        continue

    # Convert the input to a number
    number = float(user_input)

    # Update count, total, min_num, and max_num
    count += 1
    total += number

    if min_num is None or number < min_num:
        min_num = number

    if max_num is None or number > max_num:
        max_num = number

# Check if at least one number was entered
if count > 0:
    # Calculate average
    average = total / count

    # Print results
    print(f"Sum={total}")
    print(f"Average={average:.3f}")
    print(f"Minimum number={int(min_num)}")
    print(f"Maximum number={int(max_num)}")
else:
    print("No numbers entered.")
```

```
Enter a number (or 'QUIT' to finish): 4
Enter a number (or 'QUIT' to finish): 5
Enter a number (or 'QUIT' to finish): 1
Enter a number (or 'QUIT' to finish): Quit
Sum=10.0
Average=3.333
Minimum number=1
Maximum number=5
```

Q. 190

In [54]:

```
month=input()
nights=int(input())
studio_price=50.00
apartment_price=65.00
studio_rent=0.0
apartment_rent=0.0
if month=='January' or month=='February' or month=='March' or month=='April':
    studio_price=50.00
```

```
    apartment_price=60.00
    studio_rent=studio_price*nights
    apartment_rent=apartment_price*nights
    if nights>3:
        studio_rent*=0.80
    elif nights>7:
        studio_rent*=0.70
        apartment_rent*=0.90
elif month=='May' or month=='June' or month=='July' or month=='August':
    studio_price=70.00
    apartment_price=80.00
    studio_rent=studio_price*nights
    apartment_rent=apartment_price*nights
    if nights>3:
        studio_rent*=0.90

    elif nights>7:
        studio_rent*=0.80
        apartment_rent*=0.90
elif month=='September' or month=='October' or month=='November' or month=='December':
    studio_price=80.00
    apartment_price=99.00
    studio_rent=studio_price*nights
    apartment_rent=apartment_price*nights
    if nights>3:
        studio_rent*=0.95

    elif nights>7:
        studio_rent*=0.90
        apartment_rent*=0.90
print(studio_rent)
print(apartment_rent)
```

```
May
5
315.0
400.0
```

Q. 191 Write a program that enters a single digit integer number and produces all possible 6-digit numbers for which the product

of their digits is equal to the entered number. Example: "number" → 2 • 111112 → 1 *1* 1 *1* 1 *2* = *2 • 111121* → *1* 1 *1* 1 *2* 1 = 2 • 111211 → 1 *1* 1 *2* 1 *1* = *2 • 112111* → *1* 1 *2* 1 *1* 1 = 2 • 121111 → 1 *2* 1 *1* 1 *1* = *2 • 211111* → *2* 1 *1* 1 *1* 1 = 2

In [1]:
```
n = int(input('Enter a single digit number: '))
for i in range(100000, 1000000):
  m = 1
  for j in str(i):
    m *= int(j)
  if m == n:
    print(i)
```

```
Enter a single digit number: 2
111112
111121
111211
112111
121111
211111
```

Q. 192 Write a Python program that prompts the user to enter numbers and stops only when the user enters “stop”. After this,

print the minimum even, maximum even, average of even number, minimum odd, maximum odd, average of odd number from among all the numbers entered by the user. Note: You are not allowed to use any built-in structures like lists, tuples, etc. or any built-in functions like max, min, sum Example: input and output enter number or q for'stop:'-1 enter number or q for'stop:'-5 enter number or q for'stop:'9 enter number or q for'stop:'2 enter number or q for'stop:'4 enter number or q for'stop:'6 enter number or q for'stop:'stop Output: for even 6 2 4.0 (max, min, avg) for odd 9 -5 1.0 (max, min, avg)

In [12]:
```python
# Initialize variables for even numbers
# Initialize variables for even numbers
min_even = float('nan')
max_even = float('-nan')
sum_even = 0
count_even = 0

# Initialize variables for odd numbers
min_odd = float('nan')
max_odd = float('-nan')
sum_odd = 0
count_odd = 0

# Flag to control the loop
stop_flag = False

# Function to check if a string represents an integer
def is_integer(s):
    valid_digits = "0123456789"
    is_negative = False

    # Check if the first character is '-'
    if s and s[0] == '-':
        is_negative = True

    # Check if all characters are valid digits
    for char in s[is_negative:]:
        if char not in valid_digits:
            return False

    return bool(s[is_negative:])

# Loop to get user input until 'stop' is entered
```

```
while not stop_flag:
    user_input = input("Enter number or 'stop' to stop: ")

    # Check if the user wants to stop
    if user_input == 'stop':
        stop_flag = True
    else:
        # Check if the input is a valid number
        if is_integer(user_input):
            number = int(user_input)

            # Update even and odd statistics
            if number % 2 == 0:
                if count_even == 0 or number < min_even:
                    min_even = number
                if count_even == 0 or number > max_even:
                    max_even = number
                sum_even += number
                count_even += 1
            else:
                if count_odd == 0 or number < min_odd:
                    min_odd = number
                if count_odd == 0 or number > max_odd:
                    max_odd = number
                sum_odd += number
                count_odd += 1
        else:
            print("Invalid input. Please enter a valid number.")

# Calculate averages
avg_even = sum_even / count_even if count_even > 0 else 0
avg_odd = sum_odd / count_odd if count_odd > 0 else 0

# Display results
print(f"For even: {max_even} {min_even} {avg_even} (max, min, avg)")
print(f"For odd: {max_odd} {min_odd} {avg_odd} (max, min, avg)")
```

```
Enter number or 'stop' to stop: -1
Enter number or 'stop' to stop: -5
Enter number or 'stop' to stop: 9
Enter number or 'stop' to stop: 2
Enter number or 'stop' to stop: 4
Enter number or 'stop' to stop: 6
Enter number or 'stop' to stop: stop
For even: 6 2 4.0 (max, min, avg)
For odd: 9 -5 1.0 (max, min, avg)
```

Q. 235

Write your own python program for computing square roots that implements Newton's Method. Use of inbuilt function, math library or x**0.5 is not allowed. Newton Method is a category of guess-and-check approach. You first guess what the square root might be and then see how close your guess is. You can use this information to make another guess and continue guessing until you have found the square root (or a close approximation to it). Suppose x is the number we want the root of and guess is the current guessed answer. The guess can be improved by using (guess+ x/guess)/2 as the next guess. The program should -

1. Prompt the user for the value to find the square root of (x) and the number of times to improve the guess.
2. Starting with a guess value of x/2, your program should loop the specified number of times applying Newton's method and report the final value of guess.

In [90]:
```
def square_root_newton(x, num_iterations):
    # Initial guess (starting point)
    guess = x / 2.0

    # Apply Newton's Method iteratively
    for _ in range(num_iterations):
        guess = (guess + x / guess) / 2.0

    return guess

# Prompt the user for input
x = float(input("Enter the number to find the square root of: "))
num_iterations = int(input("Enter the number of iterations: "))

# Check for non-negative input
if x < 0:
    print("Cannot compute the square root of a negative number.")
else:
    # Compute and print the square root using Newton's Method
    result = square_root_newton(x, num_iterations)
    print(f"The square root of {x} is approximately {result:.6f}")
```

```
Enter the number to find the square root of: 25
Enter the number of iterations: 6
The square root of 25.0 is approximately 5.000000
```

In [ ]: