Q1) Write a program to input a decimal number then print a binary string with padding zeros until making the length 1Byte?

```
def decimal_to_binary(number):
    result = ""
    while number != 0:
        result = str(number % 2) + result
        number //= 2
    return (8-len(result))*"0"+result


x = int(input('decimal number: '))
print(decimal_to_binary(x))
```

Q2) Write a program to convert a string of characters to string of binary number? Each character should be padded to length of 1Byte.

```
def decimal_to_binary(number):
    output = ''
    while number != 0:
        output = str(number % 2) + output
        number //= 2
    return output


string_input = input("Please Enter a string:")
string_to_binary = ""
for char_value in string_input:
```

```
        binary_value = decimal_to_binary(ord(char_value))
        length_of_binary = len(binary_value)
        missed_bits = 8-length_of_binary
        padding_zeros = ""
        padding_zeros = missed_bits * "0"
        binary_value = padding_zeros + binary_value
        string_to_binary += binary_value
    print(string_to_binary)
```

Q3) Write a program to convert string of binary to decimal number?

```
 def binary_to_decimal(binary_value):
    number = 0
    i = 0
    reverse_bin = binary_value[::-1]
    for n in reverse_bin:
        if n == "1":
            number += pow(2, i)
        i += 1
    return number


binary_string = input("please Enter a binary string: ")
print(">> the decimal number is: " + str(
binary_to_decimal(binary_string)))
```

Q4) Write a program to convert each 1byte string of binary to a character? The length of string of binary may be greater than 1 byte (i.e. the output may be more than one character?

```
def binary_to_decimal(binary_value):
    number = 0
    counter = 0
    reverse_bin = binary_value[::-1]
    for bit in reverse_bin:
        number += int(bit) * pow(2, counter)
        counter += 1
    return number


binary_string = input('Please Enter a string of binary
numbers: ')
binary_char = ""
i = 0
if len(binary_string) % 8 == 0:
    for n in range(len(binary_string)//8):
        binary_char +=
chr(binary_to_decimal(binary_string[i: i+8]))
        i += 8
    print(">>the string is: " + binary_char)
else:
    print('Invalid data, please enter a correct string of
binary')
```

Q5) Write a program to check if number is prime or composite?

```
def prime_checker(number):
    is_prime = True
    for i in range(2, number):
```

```
        if number % i == 0:
            is_prime = False
            break
    return is_prime

x = int(input(">>please Enter a number: "))
if x == 0 or x == 1:
    print(str(x) + " is not prime nor composite")
elif prime_checker(x):
    print(str(x) + " is prime")
else:
    print(str(x) + " is composite")
```

Q6) Write a program to calculate greats common divisor between two numbers? Use recursion function.

```
    def gcd(a, b):
        if b == 0:
            return a
        else:
            return gcd(b, a % b)


    number1 = int(input(">>Please Enter first Number: "))
    number2 = int(input(">> please Enter second number: "))
    print("The GCD =" + str(gcd(number1, number2)))
```

Q7) Write a program to calculate greats common divisor between two numbers? Without using recursion function.

```
import termcolor


def gcd(a, b):
    if b > a:
        a, b = b, a
    while b != 0:
        q = a // b
        r = a % b
        print(termcolor.colored(str(a) + " = " +
str(q)+"(" + str(b) + ")+ " + str(r), 'blue'))
        a, b = b, a % b
    return a


c = int(input(">>Enter first number: "))
d = int(input(">>Enter second number: "))
print("GCD= " + str(gcd(c, d)))
```

Q8) Write a program to calculate the multiplication module inverse using iteration method?

```
def gcd(a, b):
    q = []
    if b > a:
        a, b = b, a
    while b != 0:
        q.append(a // b)
        a, b = b, a % b
    return q, a
```

```
def inverse(a, b):
    num = a
    mod = b
    x = [0, 1]
    y = [1, 0]
    q1, gcd1 = gcd(num, mod)
    if gcd1 == 1:
        for iteration in range(0, len(q1) - 1):
            x.append(-1 * q1[iteration] * x[iteration + 1]
                     + x[iteration])
            y.append(-1 * q1[iteration] * y[iteration + 1]
                     + y[iteration])
        return x[-1] % mod
    else:
        return False


a1 = int(input("please Enter first number: "))
b1 = int(input("please Enter second number: "))
inv = inverse(a1, b1)
if inv:
    print('inverse= ' + str(inv))
else:
    print("Inverse doesn't Exit")
```

Q9) Write a program to calculate the multiplication module inverse using extended Euclidean method?

```
def gcd(x, y):
    q1 = []
    if y > x:
        x, y = y, x
    while y != 0:
        q1.append(x // y)
        x, y = y, x % y
    return x


def inv(x, y):
    mod = y
    if gcd(x, y) == 1:
        a = 0
        b = 1
        while y % x != 0:
            q = y // x
            r = y % x
            ab = a - b * q
            print(ab)
            y, x, a, b = x, r, b, ab
        return ab % mod
    else:
        return False


number = int(input("Please Enter a Number: "))
mod1 = int(input("Please Enter a modular: "))
print("inverse = " + str(inv(number, mod1)))
```

Q10) Write a program to calculate the exponential module using Binary method?

```
def decimal_to_binary(number):
    result = ''
    while number != 0:
        result = str(number % 2) + result
        number //= 2
    return result


def modular_exponential(num, expo, mod):
    binary = decimal_to_binary(expo)
    if binary[0] == '1':
        c = num
    else:
        c = 1
    print(str(binary[0]) + "\t" + str(c) + "\t" + "----")
    for i in range(1,len(binary)):
        c = c**2 % mod
        if binary[i] == '1':
            c = c * num % mod
            print(str(binary[i]) + "\t" + str(c) + "\t" +
str(c))
        else:
            print(str(binary[i]) + "\t" + str(c) + "\t" +
"----")
    return c


print(" Finding m^e mod n")
```

```
m = int(input("Enter m= "))
e = int(input("Enter e= "))
n = int(input("Enter n= "))
print("Result= " + str(modular_exponential(m, e, n)))
```

Q11) Write a program to calculate the exponential module using Recursion method?

```
def modular_exponential(num, expo, mod):
    if expo == 0:
        return 1
    elif expo % 2 == 1:
        return modular_exponential(num, expo-1, mod) * num
% mod
    else:
        return modular_exponential(num, expo//2, mod)**2 %
mod
```

```
print(" Finding m^e mod n")
m = int(input("Enter m= "))
e = int(input("Enter e= "))
n = int(input("Enter n= "))
print("Result= " + str(modular_exponential(m, e, n)))
```

HomeWorks:
Write all programs above in Python GUI mode.