Q1)

		Remainder	
1)	$1621/2 \rightarrow 810$	1	LSB
	$810/2 \rightarrow 405$	0	
	$405/2 \rightarrow 202$	1	
	$202/2 \rightarrow 101$	0	
	$101/2 \rightarrow 50$	1	
	$50/2 \rightarrow 25$	0	
	$25/2 \rightarrow 12$	1	
	$12/2 \rightarrow 6$	0	
	$6/2 \rightarrow 3$	0	
	$3/2 \rightarrow 1$	1	
	$1/2 \rightarrow 0$	1	MSB

Thus 1621 in Unsigned integer representation is $(1621)_{10} = (11001010101)_2$

For $\frac{443}{2048}$ we can solve it in two method

method 1) Realise 2048 is 2^x where $x = 11$ by division thus 443×2^{-11} and find 443 in binary

$443/2 \rightarrow 221$	1	LSB
$221/2 \rightarrow 110$	1	
$110/2 \rightarrow 55$	0	
$55/2 \rightarrow 27$	1	
$27/2 \rightarrow 13$	1	
$13/2 \rightarrow 6$	1	
$6/2 \rightarrow 3$	0	
$3/2 \rightarrow 1$	1	
$1/2 \rightarrow 0$	1	MSB

Thus, $(110111011)_2 \times 2^{-11}$

$\left(\frac{443}{2048}\right)_{10} = (0.00110111011)_2$

Method 2) $\frac{443}{2048}$ since its less than 1 I multiply by 2 to find its binary representation

$$\frac{443}{2048} \times 2 \longrightarrow \frac{886}{2048} \quad 0 \quad \text{MSB}$$

$$\frac{886}{2048} \times 2 \longrightarrow \frac{1772}{2048} \quad 0$$

$$\frac{1772}{2048} \times 2 \longrightarrow \frac{3544}{2048} \quad 1$$

$$\frac{1496}{2048} \times 2 \longrightarrow \frac{2992}{2048} \quad 1$$

$$\frac{944}{2048} \times 2 \longrightarrow \frac{1888}{2048} \quad 0$$

$$\frac{1888}{2048} \times 2 \longrightarrow \frac{3776}{2048} \quad 1$$

$$\frac{1728}{2048} \times 2 \longrightarrow \frac{3456}{2048} \quad 1$$

$$\frac{1408}{2048} \times 2 \longrightarrow \frac{2816}{2048} \quad 1$$

$$\frac{768}{2048} \times 2 \longrightarrow \frac{1536}{2048} \quad 0$$

$$\frac{1536}{2048} \times 2 \longrightarrow \frac{3072}{2048} \quad 1$$

$$\frac{1024}{2048} \times 2 \longrightarrow \frac{2048}{2048} \quad 1 \quad \text{LSB}$$

Thus $\left(\frac{443}{2048}\right)_{10} = (0.00110111011)_2$

Q2)

```python
####################################################
## Abdullah Jafar Mansour Shamout
## 150200919
####################################################

# since nothing was mentioned about the standard to represent the numbers in, for -ve numbers I will just take them with their absoulute value as input
# and in the output I will put a -ve sign, I can also use a bit to represent it but i didnt want their to be confusion so i will just add the sign

def question2(x):
    # Here I am extracting the integer part of the rational number x by casting it to an int and truncating the decimal values
    whole_part = abs(int(x))

    # Here I extract the fractional part
    fractional_part = abs(x) - whole_part

    # Here I convert the integer part to its binary representation by using a loop
    # First I start with an empty string
    Bwhole_part = ''
    # I check for if the whole part is 0
    if whole_part == 0:
        Bwhole_part = '0'
    while whole_part > 0:   # Here I go through a loop while the whole part is not 0
        Bwhole_part = str(whole_part % 2) + Bwhole_part     # I use modulus 2 to get the remainder, convert it to a string and concatenate it to the le
        whole_part //= 2    # Here I floor divide the number since I dont need the remainder as I calculated it

    # Here I convert the fractional part to its binary representation
    # First I start with an empty string
    Bfractional_part = ''
    if fractional_part == 0:
        Bfractional_part = '0'
    while fractional_part > 0.000001:  # Just like the method used in class I am multiplying by 2 and taking the whole number part for my binary repres
        fractional_part *= 2
        int_part = int(fractional_part)
        Bfractional_part += str(int_part) # Here I am concatinating the result I got to my representation
        fractional_part -= int_part # Here I remove the integer part from the fractional part to continue with the calculation

    # Here I combine both parts to get my full representation
    binary = Bwhole_part + '.' + Bfractional_part

    # Here I add a -ve sign to the representation if the number is -ve
    if x < 0:
        binary = '-' + binary

    return binary   #finally I return the binary representation

# Some examples as requested in the question
# Example 1

print("#########Here are some examples##########")

x = 12.125
binary = question2(x)
print(binary)

# Example 2
x = 0.25
binary = question2(x)
print(binary)

# Example 3
x = 32.625
binary = question2(x)
print(binary)

# Example 4
x = 2.1
binary = question2(x)
print(binary)

# Example 5
x = -12.125
binary = question2(x)
print(binary)

print("#########################################")

x = float(input("Enter the value of the rational number you want to test for in decimal format: "))
binary = question2(x)
print(binary)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\abdul\Desktop\3rd semester\blg223\Recitation_5> & C:/Users/abdul/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/abdul/Desktop/numerical_hw/Question2
.py
#########Here are some examples##########
1100.001
0.01
100000.101
10.0001100110011001100110011001100110011001101
-1100.001
#########################################
Enter the value of the rational number you want to test for in decimal format: -32.0625
-100000.0001
PS C:\Users\abdul\Desktop\3rd semester\blg223\Recitation_5>
```

Q3)

3) Method of Bisection for $17^{\frac{1}{3}}$. We use this Method to approximate for roots, so I assumed a function, $f(x) = x^3 - 17$. that would have a root at $x = 17^{\frac{1}{3}}$, by picking two values $x = 2, 3$ $2^3 = 8$ which is less than 17 → −ve value and $3^3 = 27$ → +ve value.

① $2^3 - 17 = -9 < 0$, $3^3 - 17 = 10 > 0$ → $\frac{2+3}{2} = 2.5$ ②

$(2.5)^3 - 17 = -1.375$ ④

③ $\frac{2.5+3}{2} = 2.75$ → $(2.75)^3 - 17 = 3.796875$ ⑧

⑤ $\frac{2.75 + 2.5}{2} = 2.625 \rightarrow (2.625)^3 - 17 = 1.08789...$

⑥ $\frac{2.625 + 2.5}{2} = 2.5625$ $\boxed{\text{I will stop here}}$

$17^{1/3} \simeq 2.571281591$ our Approximation is 2.5625

Absolute error : $|17^{1/3} - 2.5625| \simeq 8.78 \times 10^{-3}$

Relative error : $\frac{|17^{1/3} - 2.5625|}{17^{1/3}} \simeq 3.42 \times 10^{-3}$

Using Taylor Expansion for $17^{1/3}$. Assume a function $f(x) = x^{1/3}$ an easy number to take it's cubic root would be 8. Thus, by using taylor series expansion of $f(x)$ around 8, we can find an easy approximation

General rule:

$$f(x) = f(c) + f'(c)(x-c) + \frac{f''(c)(x-c)^2}{2!} + \frac{f'''(c)(x-c)^3}{3!} \cdots$$

using $c = 8$ and approximating for $x = 17$ we get:

$$f(17) \simeq f(8) + f'(8)(17-8) + \frac{f''(8)(17-8)^2}{2!} + \frac{f'''(8)(17-8)^3}{3!} \cdots$$

I will stop approximating at the $f'''(c)$ term.

$$f(x) = x^{1/3},\ f'(x) = \frac{1}{3}x^{-\frac{2}{3}},\ f''(x) = -\frac{2}{9}x^{-\frac{5}{3}},\ f'''(x) = \frac{10}{27}x^{-8/3}$$

$$f(17) \simeq 2 + \left(\frac{1}{3}\right)\left(\frac{1}{8^{2/3}}\right)(9) + \left(-\frac{2}{9}\right)\left(\frac{1}{8^{5/3}}\right)\left(\frac{1}{2!}\right)(9)^2 + \left(\frac{10}{27}\right)\left(\frac{1}{8^{8/3}}\right)\left(\frac{1}{3!}\right)(9)^3$$

$$f(17) \simeq 2 + \frac{3}{4} + \left(-\frac{9}{32}\right) + \frac{45}{256}$$

$f(17) \simeq 2.645$ while $17^{1/3} = 2.57128\ldots$

Absolute Error $|17^{1/3} - 2.645| = 0.074$

Relative Error $\frac{|17^{1/3} - 2.645|}{17^{1/3}} = 0.029$

Q4)

Question4.py

C: > Users > abdul > Desktop > numerical_hw > Question4.py > ...

```python
##################################################
## Abdullah Jafar Mansour Shamout
## 150200919
##################################################

def question5(a, epsilon):
    # Since "a" can be either positive or negative I need to initialize the intervals of testing accordingly
    #if a >= 0 then the minimum value for a^(1/5) is 0 thus our minimum interval will start at 0 and our maximum will start at "a" since a^(1/5) can be a maximum of "a" when a=1
    if a >= 0:
        min = 0
        max = a
    else:
        min = a
        max = 0

    mid = (min + max) / 2 # Here I am using the bisection method to initialize the mid value
    while abs(a - mid**5) > epsilon: # Here I started a loop that would update min/max values with newly found mid untill we get an error less than epsilon
        if mid ** 5 < a:
            min = mid   # Here if mid^5 is less than the "a" we need that, means that mid is a closer minimum to our "a" value so we update min
        else:
            max = mid   # Here if mid^5 is greater than the "a" we need that, means that mid is a closer maximum to our "a" value so we update max
        mid = (min + max) / 2 # Here we calculate a new min and check it again in the loop

    return mid

print("#########Here are some examples##########")

# Some examples as requested in the question
# Example 1
a = 32
epsilon = 0.01
x = question5(a,epsilon)
print(x)

# Example 1 negative case
a = -32
epsilon = 0.01
x = question5(a,epsilon)
print(x)

# Example 2
a = 243
epsilon = 0.001
x = question5(a,epsilon)
print(x)

# Example 2 negative case
a = -243
epsilon = 0.001
x = question5(a,epsilon)
print(x)

print("##########################################")
print("Enter the values you want to test for 'a' and epsilon")
a = input("Enter the value for a: ")
epsilon = input("Enter the value for epsilon: ")
x = question5(float(a),abs(float(epsilon)))
print(x)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\abdul> & C:/Users/abdul/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/abdul/Desktop/numerical_hw/Question4.py
#########Here are some examples##########
2.0
-2.0
2.9999982118606567
-2.9999982118606567
##########################################
Enter the values you want to test for 'a' and epsilon
Enter the value for a: 64
Enter the value for epsilon: 0.0001
2.297396659851074
PS C:\Users\abdul>
```

powershell
Python