Câu 1

```
def euclid_algorithm(u, v):
  u1, u2, u3 = 1, 0, u
  v1, v2, v3 = 0, 1, v

  while v3 != 0:
    q = u3 // v3
    t1, t2, t3 = u1 - q*v1, u2 - q*v2, u3 - q*v3
    u1, u2, u3 = v1, v2, v3
    v1, v2, v3= t1, t2, t3

  return u1, u2, u3

u = 42
v = 56
print(euclid_algorithm(u, v))
```

```
PS D:\HCMUS\Computer Science> & "D:/Python Anaconda/anaconda3/python.exe" "d:/HCMUS/Co
mputer Science/Số học và thuật toán/Requirement 3/Assignment 3.py"
(-1, 1, 14)
PS D:\HCMUS\Computer Science>
```

Câu 2

```
import math
#Return the number of digits
def count_digits(number):
    count = len(str(number))
    return count

#Split the number in half
def split_number(number):
    n = number
    digits = count_digits(n)
    if (digits%2) != 0: digits -= 1
    b = n%(10**(digits//2))
    a = (n-b)//(10**(digits//2))
    return a,b

#Recursive algorithm
```

```
def multiply(x,y, count):

    #Get both digit counts, i probably have to do it later
    n1 = count_digits(x)
    n2 = count_digits(y)
    #Get smallest digit of the two numbers to check if one is single digit
    n = min(n1,n2)
    #If the smallest number is one digit multipy the two numbers together
    if (n == 1):
        return x*y
    #If one is bigger than than other digit count-wise start recursion
    if (n != 1):
        ##padding for abritary sizes of two number sets##

        padding = 0
        #if digit count if different, pad the smaller number with zeros
        if( n1 != n2 ):
            #record number of zeros needed to revert later
            padding = abs(n1 - n2)

            #pad the appropriate number
            if n1 < n2:
                x = x*10**padding
            else:
                y = y*10**padding

        #split number sets up
        a, b = split_number(x)
        c, d = split_number(y)

        #recursively call them until they are digit to multiply
        #find the products of ac, ad, bc, bd
        ac = multiply(a,c,count)
        ad = multiply(a,d,count)
        bc = multiply(b,c,count)
        bd = multiply(b,d,count)

        #Find the largest number in the orgnal set
        #this is to keep consistency and have the correct N digit to raise 10 to
        n = max(n1,n2)


        #if its an odd number of digits
        if n%2:
            n = n -1
```

```
        nhalf = n//2


        results = (10**(n))*ac + (10**(nhalf))*(ad+bc) + bd

    #if padded divided by the padding to take it away
        if padding > 0:
            return results//10**padding

        return int(results)

        ##Python floating point calculations are way off
        ##I had to make sure I stayed with ints and no .0
        ##were trailing. I do so with int() calles and
        ## double divide //. If not then the results
        ## were off with bigger numbers

def main(x,y):
    results = multiply(x,y,0)
    return results

#Global for test running
X =
12564564564456456456564564564222225554444444444444444444444444444444444444444444444
4446216556465456164564565644 00
Y =
15125645645644564564565645645642222255544444444444444444444444444444444444444444444
44444462165564654561645645656440 0
global Results

def test():
    global Results
    Results = main(X,Y)


def test2():
    return X * Y

if __name__ == "__main__":

    print("Results: " + str(Results))

    #Double check results, throw a fit if they are wrong
    if ( X*Y != Results):
        print("Something went wrong, results are not what was expected")
```

```
        print(str(X*Y))
        assert(X*Y == Results)
```

```
PS D:\HCMUS\Computer Science> & "D:/Python Anaconda/anaconda3/python.exe" "d:/HCMUS/Co
mputer Science/Số học và thuật toán/Requirement 3/Nguyen.py"
Results: 190047151293790795481813754650335111647284973615065374570958918896103814740753086419780113416386331245035378555611865668064243304642575045854299499269135802469135802469135833872945952611567527557343601422835935983851347360000
PS D:\HCMUS\Computer Science>
```