CODE MẪU BÀI TẬP TUẦN 6

Bài 1:

Cho các hàm số sau:

$$f(x)=\begin{cases}-2(x-3) & \text{khi} \quad -1\le x\le 1\\ \sqrt{x^2-1} & \text{khi} \quad x>1\end{cases} \qquad g(x)=\begin{cases}x^2+1\ khi\ x>2\\ 2x-1\ khi\ -2\le x\le 2\\ 6-5x\ khi\ x<-2\end{cases}$$

Viết chương trình cho phép thực hiện các công việc sau:

- Viết hàm f(x) và g(x)
- Nhập một số thực x từ bàn phím. Tính và hiển thị giá trị của các hàm f(x) và g(x) tương ứng (Lấy 2 chữ số sau dấu phẩy)

Input: Số thực x

Output:

- Dòng 1: Giá trị hàm f(x)
- Dòng 2: Giá trị hàm g(x)

For example:

Input	Output
3.5	3.35 13.25
2	1.73 3.00

```python
# Code: Minh Khiet
import math
# tạo hàm f với giá trị truyền vào tham số x
def f(x):
    if x > 1: # trường hợp x > 1
        return math.sqrt(x**2 - 1)
    elif x >= -1: # trường hợp -1 <= x <= 1
        return (-2) * (x -3)
    else: # trường hợp x < -1
        return 0

# tạo hàm g với giá trị truyền vào tham số x
def g(x):
    if x > 2: # trường hợp x > 2
        return x**2 + 1
    elif x >= -2: # trường hợp -2 <= x <= 2
        return 2 * x - 1
    else: # trường hợp x < -2
        return 6 - 5 * x

# nhập giá trị cần truyền vào hàm
x = float(input())
# in ra hàm kết quả mà hàm f(x) trả về
print(f"{f(x):.2f}")
# in ra hàm kết quả mà hàm g(x) trả về
print(f"{g(x):.2f}")
```

Bài 2:

Cho các hàm số sau:

$$f(x,y) = x^2 + xy + y^2 - 2x - y \qquad f(x,y,z) = xyz + \frac{x}{y^z}$$

Viết chương trình cho phép thực hiện các công việc sau:

- Viết hàm f(x, y) và f(x, y, z)
- Nhập các số thực x, y từ bàn phím. Tính và hiển thị giá trị của các hàm tương ứng (Lấy 2 chữ số sau dấu phẩy). Trường hợp x, y làm cho hàm vô nghĩa, hiển thị chuỗi "N/A".

Input:

- Dòng 1: Số thực x
- Dòng 2: Số thực y
- Dòng 3: Số thực z

Output:

- Dòng 1: Giá trị hàm f(x, y)
- Dòng 2: Giá trị hàm f(x, y, z)

For example:

Input	Output
3.5 2 1	14.25 8.75
2 0 1	0.00 N/A

```
# Code: Trịnh Hưng
#tạo hàm fxy để tính giá trị hàm f(x,y)
def fxy(x,y) :
    print(f'{x**2 +x*y +y**2 -2*x - y:.2f}')

#Tạo hàm fxyz để tính giá trị hàm f(x,y,z)
def fxyz(x,y,z):
    #Kiểm tra giá trị của y thỏa mãn biểu thức toán học hay không
    if y == 0 :
        print('N/A')
    else :
        print(f'{x*y*z+x/(y**z):.2f}')
x = float(input())
y = float(input())
z = float(input())

#Gọi hàm fxy và chuyền tham số x,y để suất giá trị của f(x,y) ra màn hình
fxy(x,y)

#gọi hàm fxyz và chuyền tham số x,y,z để suất giá trị của hàm f(x,y,z) ra màn hình
fxyz(x,y,z)
```

Bài 3:

Hàm S được tính theo công thức sau đây:

$$S(x, n) = x + x^2 + x^3 + \ldots + x^n$$

Viết chương trình cho phép thực hiện các công việc sau đây:

- Viết hàm tính S theo công thức trên
- Nhập một số thực x và số nguyên n từ bàn phím. Tính và hiển thị giá trị của S tương ứng (Lấy 3 chữ số sau dấu phẩy thập phân)

Input:

- Dòng 1: Số thực x
- Dòng 2: Số nguyên n

Output: Giá trị của S tương ứng

For example:

Input	Output
2 1	2.000
3.5 5	733.906

```
# Code: Minh Khiet
# tạo hàm S với tham số truyền vào là x và n
def S(x,n):
    # khai báo biến s để lưu trữ giá trị tổng x mũ ..
    s = 0
    # tạo vòng lặp for chạy từ giá trị 1 đến n
    for i in range (1,n+1):
        s += x**i # cộng dồn biến s qua các lần lặp
    return s

# nhập số thực x
x = float(input())
# nhập số nguyên m
n = int(input())
# in ra màn hình giá trị trả về của hàm S với kết quả làm tròn số thập phân thứ 3
print(f"{S(x,n):.3f}")
```

Bài 4:

Số ***nguyên tố*** là số chỉ có ước là 1 và chính nó. Viết chương trình nhập cho phép thực hiện các công việc sau:

- Viết một hàm kiểm tra một số nguyên dương n có phải là số nguyên tố hay không?
- Viết một hàm kiểm tra một số nguyên dương n có nằm trong khoảng [a, b] nào đó hay không?
- Nhập hai số nguyên dương m, n từ bàn phím (500<m, n<100000). Đếm và hiển thị các tất cả các số nguyên tố nằm trong khoảng (m, n). Nếu m hoặc n không nằm trong khoảng (500, 100000), hiển thị chữ "N/A"

Input:

- Dòng 1: Số nguyên dương m
- Dòng 2: Số nguyên dương m

Output: Số lượng các số nguyên tố nằm trong khoảng (m, n)
For example:

Input	Output
100 9000	N/A
600 1200	87
550 90000	8612

```
# Code: Trịnh Hưng
#Tạo hàm isPrime(n) để kiểm tra tham số n chuyền vào có phải là số nguyên tố hay không.
def isPrime(n) :
    if n < 2 : #Vì các số nguyên nhỏ hơn 2 đều là hợp số(Không xét các số âm)
        return False
    #Kiểm tra trong khoảng từ 2 đến phần nguyên của căn bậc 2 của n
    for i in range(2,int(n**0.5)+1) :
        if n % i == 0 : #Nếu xuất hiện ước của n trong khoảng kiểm tra t sẽ trả về giá trị False và số n không là số nguyên tố
            return False
    return True #Nếu tất cả đều thỏa mãn thì trả về giá trị True cho hàm và số n là số nguyên tố

m = int (input())
n = int(input())

#Nếu m,n không thỏa mãn điều kiện đề bài ra thì print('N/A')
if m < 500 or n > 100000 :
    print('N/A')
else :
    #Khởi tạo 1 biến để đếm số lượng các số nguyên tố trong khoảng [m,n]
    dem = 0
    for i in range(m,n+1) :
        if isPrime(i) == True :
            dem += 1 #Nếu i là số nguyên tố thì dem tăng lên 1
    print(dem)
```

Bài 5:

Số ***hoàn chỉnh*** là số bằng tổng tất cả các ước nhỏ hơn nó, ví dụ 6 = 1 + 2 + 3 là một số hoàn chỉnh. Viết chương trình cho phép thực hiện các công việc sau:

- Viết một hàm kiểm tra một số nguyên dương n có phải là số hoàn chỉnh hay không?
- Viết một hàm kiểm tra một số nguyên dương n có nằm trong khoảng [a, b] nào đó hay không?
- Nhập một số nguyên dương n từ bàn phím (1<n<100000). Tính và hiển thị theo thứ tự tăng dần tất cả các số hoàn chỉnh nhỏ hơn hoặc bằng n. Nếu n không nằm trong khoảng (1, 100000), hiển thị chữ "N/A"

Input:

- Dòng 1: Số nguyên dương n

Output: Các số hoàn chỉnh nhỏ hơn hoặc bằng n
For example:

Input	Output
1	N/A
1000	6 28 496
555	6 28 496

```python
# Code: Minh Khiet
# tạo hàm isPerfectNumber Kiểm tra một số nguyên dương n có phải là số hoàn chỉnh hay không?
def isPerfectNumber(n):
    sum = 0 # tạo biến sum để tính tổng các số nguyên tố
    for i in range(1,n):
        if n%i == 0: # kiểm tra số nguyên tố
            sum += i # tổng các số nguyên tố
    return sum == n # kiểm tra nếu sum = n thì hàm trả về true ngược lại thì trả về false

# tạo hàm isInRange kiểm tra một số nguyên dương n có nằm trong khoảng [a, b] nào đó hay không?
def isInRange(n, a, b):
    return n >= a and n <= b

# tạo hàm displayPerfectNumber Tìm và hiển thị theo thứ tự tăng dần tất cả các số hoàn chỉnh nhỏ hơn hoặc bằng n.
def displayPerfectNumber(n):

    result = ""     # tạo biến result dưới dạng chuỗi để hiển thị các số hoàn chỉnh trên cùng 1 dòng
    for i in range(1, n):
        if isPerfectNumber(i):

            result += str(i) + " "      # thêm các số hoàn chỉnh vào chuỗi
    if result == "":    # kiểm tra nếu chuỗi rỗng thì in ra "N/A"
        print("N/A")
    else:
        print(result) # hiển thị chuỗi số hoàn chỉnh

# nhập số nguyên n
n = int(input())
# kiểm n có thuộc khoảng (1,100000) hay không
if isInRange(n, 1, 100000):
    displayPerfectNumber(n) # hiển thị danh sách số hoàn chỉnh
else: # nếu n không nằm trong khoảng trên
    print("N/A")
```

Bài 6:

Số ***đối xứng*** là số viết từ trái sang phải hay viết từ phải sang trái là như nhau, ví dụ 12321 là một số đối xứng. Viết chương trình cho phép thực hiện các công việc sau:

- Viết một hàm kiểm tra một số nguyên dương n có phải là số đối xứng hay không?
- Nhập 2 số nguyên dương n, m từ bàn phím (100 < n, m < 100000). Đếm và hiển thị các số đối xứng trong đoạn [n, m]. Nếu n hoặc m không thỏa mãn điều kiện thì hiển thị chữ "N/A".

Input:

- Dòng 1: Số nguyên dương n
- Dòng 2: Số nguyên dương m

Output: Số lượng các số đối xứng trong đoạn [n, m]
For example:

Input	Output
100 1000	N/A
101 1200	92

```
# Code:Trịnh Hưng
#Tạo hàm check_palindrome(n) để kiểm tra tham số n chuyền vào có phải là số đối xứng hay không.
def check_palindrome(n) :
    m = n #Khởi tạo biến m để lưu giá trị ban đầu của số n
    #Chúng ta sẽ tận dụng phép chia lấy nguyên và chia lấy dư để viết ngược lại số n và lưu lại vào biến tmp
    tmp = 0
    while n > 0 :
        tmp = tmp*10 + n%10
        #Nếu số n>0 chúng ta sẽ thực hiện n%10 để lấy phần tử cuối cùng của n rồi thêm phần tử ấy vào sau tmp
        n = n//10 #Chúng ta bỏ phần tử cuối cùng của n đi bằng n//=10

        '''Ví dụ :
            n = 123
            tmp = 0
            1: n = 123 > 0 :
                n%10 = 3
                tmp*10 + n%10 = 3
                n//10 = 12
            n = 12
            tmp 3
            2: n = 12 > 0 :
                n%10 = 2
                tmp*10 + n%10 = 32
                n//10 = 1
            n = 1
            tmp 32
            3: n = 1 > 0 :
                n%10 = 1
                tmp*10 + n%10 = 321
                n//10 = 0
            n = 0
            tmp 321(Hoàn thành việc viết ngược lại số n)
        '''
    #Vì n sau vong lặp while sẽ về 0 nên chúng ta so sánh tmp với m
    if tmp == m :
        return True #Nếu tmp == m thì số là số đối xứng
    else : return False
```

```
n = int (input())
m = int(input())

#Nếu n,m không thỏa mãn điều kiện đề bài ra thì print('N/A')
if n <= 100 or m >= 100000 :
    print('N/A')
else :
    dem = 0 #Khởi tạo 1 biến để đếm số lượng các số đối xứng trong khoảng
[m,n]
    for i in range(n,m+1) :
        if check_palindrome(i) == True :
            dem += 1 #Nếu i là số đối xứng thì dem tăng lên 1
    print(dem)
```

Bài 7:

Viết chương trình thực hiện các công việc sau đây:

- Viết một hàm tìm UCLN của hai số nguyên a, b
- Nhập 5 số nguyên dương từ bàn phím. Tìm UCLN của các số này.

Input:

- Dòng 1: Số nguyên dương thứ 1
- Dòng 2: Số nguyên dương thứ 2
- Dòng 3: Số nguyên dương thứ 3
- Dòng 4: Số nguyên dương thứ 4
- Dòng 5: Số nguyên dương thứ 5

Output: UCLN của 5 số nguyên dương

For example:

Input	Output
4 8 4 24 6	2

```
# Code:Minh Khiet
# hàm tìm UCLN của hai số nguyên
def UCLN(a, b):  # Khai báo hàm ucln với hai tham số đầu vào là a và b
    while a*b != 0:  # Lặp lại cho đến khi a hoặc b bằng 0
        # Tính phép chia lấy dư của a và b, sau đó gán kết quả vào biến r
        r = a % b
        a = b  # Gán giá trị của b cho a
        b = r  # Gán giá trị của r (phép chia lấy dư) cho b
    return a+b  # Trả về giá trị của a, đó chính là ước chung lớn nhất của a
và b

# nhập 5 số nguyên dương từ bàn phím
a = int(input()) # Nhập số nguyên dương thứ nhất
b = int(input()) # Nhập số nguyên dương thứ hai
c = int(input()) # Nhập số nguyên dương thứ ba
d = int(input()) # Nhập số nguyên dương thứ tư
e = int(input()) # Nhập số nguyên dương thứ năm
```

```
# tìm UCLN của 5 số đó bằng cách tìm UCLN lần lượt của a và b, a và c, a và
d, a và e,
# b và c, b và d, b và e, c và d, c và e, và cuối cùng là d và e
UCLN1 = UCLN(a, b)
UCLN2 = UCLN(a, c)
UCLN3 = UCLN(a, d)
UCLN4 = UCLN(a, e)
UCLN5 = UCLN(b, c)
UCLN6 = UCLN(b, d)
UCLN7 = UCLN(b, e)
UCLN8 = UCLN(c, d)
UCLN9 = UCLN(c, e)
UCLN10 = UCLN(d, e)

# tìm UCLN của 5 số
result = UCLN(UCLN1, UCLN2)
result = UCLN(result, UCLN3)
result = UCLN(result, UCLN4)
result = UCLN(result, UCLN5)
result = UCLN(result, UCLN6)
result = UCLN(result, UCLN7)
result = UCLN(result, UCLN8)
result = UCLN(result, UCLN9)
result = UCLN(result, UCLN10)

# hiển thị kết quả
print(result)
```

Bài 8:

Viết chương trình cho phép thực hiện các công việc sau đây:

- Viết một hàm kiểm tra một số nguyên dương n có nằm trong khoảng [a, b] nào đó hay không?
- Nhập một số nguyên dương từ bàn phím (1<n<10000). Sử dụng kĩ thuật đệ qui để tính tổng các số nguyên dương nhỏ hơn hoặc bằng n. Nếu n không nằm trong khoảng đã cho, hiển thị chữ "N/A".

Input:

- Dòng 1: Số nguyên dương n

Output: Tổng các số nguyên dương nhỏ hơn hoặc bằng n

For example:

Input	Output
0	N/A
10	55

```python
# Code:Trịnh Hưng
#hàm check để kiểm tra n thuộc khoảng như đề bài đã cho
def check(n):
    if n<1 or n>10000 :
        return False
    return True

#hàm sum đệ quy với n == 1 thì dừng lại và cộng tổng các giá trị của n
def sum(n):
       if n == 1 : return 1
       return n + sum(n-1)

n = int(input())
if check(n) :
    print(sum(n))
else : print('N/A')
```

Bài 9:

Viết chương trình cho phép thực hiện các công việc sau đây:

- Viết một hàm chuyển đổi một số nhị phân thành số thập phân?
- Viết một hàm kiểm tra một số có phải là số nhị phân hay không?
- Nhập vào hai số nhị phân. Kiểm tra xem hai số đó có phải là số nhị phân hay không? Nếu không, hiển thị chữ "N/A". Ngược lại, tính và hiển thị tổng của hai số đó dưới dạng số thập phân.

Input:

- Dòng 1: Số nhị phân thứ nhất
- Dòng 2: Số nhị phân thứ 2

Output: Theo ví dụ sau

For example:

Input	Output
12 1011	N/A
1010 1100	22

```
# Code: Minh Khiet
# hàm chuyển đổi một số nhị phân thành số thập phân
def binary_to_decimal(binary):
    decimal = 0  # Khởi tạo biến decimal là 0, biến này sẽ dùng để tính giá trị thập phân tương ứng với số nhị phân đưa vào
    for digit in binary:  # Duyệt qua từng chữ số của số nhị phân đưa vào
        decimal = decimal*2 + int(digit)  # Chuyển đổi số nhị phân sang thập phân, bằng cách nhân decimal cho 2 và cộng với giá trị mới được tính dựa trên chữ số tiếp theo của số nhị phân, được đưa vào qua đối số 'digit'
    return decimal  # Sau khi duyệt hết tất cả các chữ số của số nhị phân, đổi được thành số thập phân tương ứng, hàm trả về giá trị số thập phân này


# hàm kiểm tra một số có phải là số nhị phân hay không
def is_binary(num):
    for digit in num:  # Duyệt qua các chữ số trong num (số đưa vào kiểm tra)
        if digit != '0' and digit != '1':  # Nếu chữ số đang xét không phải là '0' hoặc '1'
            return False  # Hàm trả về giá trị False (số không phải là số nhị phân)
    return True  # Nếu hàm chạy được đến đây, tức là tất cả các chữ số trong num đều là 0 hoặc 1, do đó hàm trả về giá trị True (số đưa vào là số nhị phân)

# nhập hai số nhị phân từ bàn phím
binary1 = input() # Nhập số nhị phân thứ nhất
binary2 = input() # Nhập số nhị phân thứ hai

# kiểm tra xem hai số nhập vào có phải là số nhị phân hay không
if not is_binary(binary1) or not is_binary(binary2):
    print("N/A")
else:
    # chuyển đổi hai số nhị phân thành số thập phân
    decimal1 = binary_to_decimal(binary1)
    decimal2 = binary_to_decimal(binary2)

    # tính tổng của hai số thập phân
    decimal_sum = decimal1 + decimal2
```

```
    # hiển thị kết quả theo dạng số thập phân
    print(decimal_sum)
```

Bài 10:

Hàm sin(x) được tính theo công thức sau đây:

$$sinx = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots + (-1)^{m-1}\frac{x^{2m-1}}{(2m-1)!}$$

Viết chương trình cho phép thực hiện các công việc sau đây:

- Viết hàm tính sin(x) theo công thức trên
- Nhập một số thực x, số nguyên m từ bàn phím. Tính và hiển thị giá trị của sin(x) tương ứng (Lấy 3 chữ số sau dấu phẩy thập phân).

Input:

- Dòng 1: Số thực x
- Dòng 2: Số nguyên m

Output: Giá trị của sin(x)

For example:

Input	Output
2 1	2.000
3.5 5	-0.328

```python
# Code: Trịnh Hưng
#hàm sinx theo yêu cầu đề bài
def sinx(x, m):
    res = 0 # biến cộng dồn kết quả
    d = 1 # tính 1^(2*i - 1): ta chỉ cần nhân - 1 để d luân phiên dấu +, -
    tuso = x # biến tính tử số của các phân số
    # ta nhân x ^ 2(x * x) để tính các tử số tiếp theo
    mauso = 1 # biến tính mẫu số của các phân số
    # ta nhân với (i + 1)*(i + 2) để tính i! tiếp theo với các i lẻ
    #ví dụ: i == 1: từ 1! muốn tính được 3! ta cần res = 1! * 2 * 3 hay res = res * (i + 1)*(i + 2)
    for i in range(1, 2*m, 2):
        res += d * tuso/mauso
        tuso *= x*x
        mauso *= (i + 1) * (i + 2)
        d *= -1
    return res

x = float(input())
m = int(input())
print(f'{sinx(x, m):.3f}')
```

End!