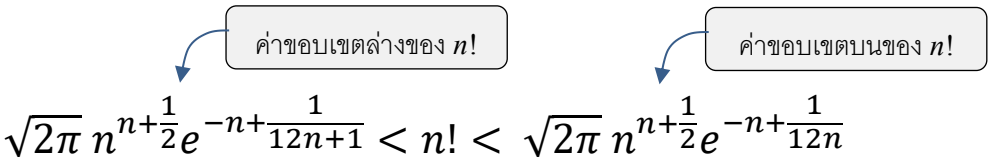
การประมาณค่าของ n !

ถ้าอยากรู้ว่า 100 ! มีค่าใหญ่ขนาดไหน ก็คงต้องคิดถึง Stirling's approximation ที่คำนวณช่วงของค่า $n!$ ด้วยสูตรข้างล่างนี้



งานของคุณ

เขียนโปรแกรมรับจำนวนเต็ม n เพื่อแสดงขอบเขตล่างและบนของการประมาณค่าของ $n!$ จากสูตรข้างบนนี้

ข้อมูลนำเข้า

จำนวนเต็ม n

ข้อมูลส่งออก

ค่าขอบเขตล่าง และค่าขอบเขตบนของ $n!$

ตัวอย่าง

input (จากแป้นพิมพ์)	**output (ทางจอภาพ)**
1	0.9958701614627972 1.0022744491822266
5	119.9698539592089 120.00263708619698
50	3.0414009534599554e+64 3.0414093877504934e+64
100	9.332615094728998e+157 9.332621570317666e+157

สูตรหารากของสมการกำลังสอง

รากจริงของสมการ $ax^2 + bx + c = 0$ คือ

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

โปรแกรมที่ต้องเขียน

ให้เขียนโปรแกรมรับจำนวนจริง a, b และ c เพื่อคำนวณและแสดงรากจริงของสมการ $ax^2 + bx + c = 0$

ข้อมูลนำเข้า

จำนวนจริง a, b และ c บรรทัดละค่า โดยสมการ $ax^2 + bx + c = 0$ ที่ให้มานี้ จะมีรากเป็นค่าจริงสองค่าที่ต่างกันแน่นอน

ข้อมูลส่งออก

รากจริงทั้งสองค่าของสมการ $ax^2 + bx + c = 0$ โดย

- แสดงราก x_1 แล้วตามด้วยราก x_2
- มีเลขหลังจุดทศนิยม 3 ตำแหน่ง (ใช้ฟังก์ชัน `round` เช่น `round(2/3, 3)` จะได้ `0.667`)

ตัวอย่าง

input (จากแป้นพิมพ์)	**output (ทางจอภาพ)**

`1.0`
`-5.0`
`6.0`
	`2.0 3.0`

`1.0`
`-1`
`-42`
	`-6.0 7.0`

`6`
`-4.0`
`-12`
	`-1.12 1.786`

`20.0`
`-50.5`
`-21.2`
	`-0.367 2.892`

Expression

จงเขียนโปรแกรมที่แสดงผลลัพธ์ของการคำนวณข้างล่างนี้

$$\frac{\pi - \frac{10!}{8^8} + (\log_e 9.7)^{\frac{7}{\sqrt{71}}} - \sin(40^\circ)}{(1.2)^{\sqrt[3]{2.3}}}$$

ข้อมูลนำเข้า

ไม่มี

ข้อมูลส่งออก

แสดงผลลัพธ์ของการคำนวณในโจทย์ (ประมาณ 3.2 กว่า ๆ)

โดยแสดงเลขหลังจุดทศนิยม 6 ตำแหน่ง (ใช้ฟังก์ชัน `round` เช่น `round(2/3, 3)` จะได้ `0.667`)

ตัวอย่าง

ไม่มี

Body_Surface_Area

พื้นที่ผิวกาย (body surface area) เป็นค่าหนึ่งที่มักในวงการแพทย์เพื่อกำหนดปริมาณยาที่ใช้ในการรักษา มีสูตรในการประมาณพื้นที่ผิวกายหลายสูตรดังแสดงข้างล่างนี้ (W คือน้ำหนัก หน่วยเป็นกิโลกรัม H คือความสูง หน่วยเป็นเซนติเมตร)

สูตรของ Mosteller	$\frac{\sqrt{W \times H}}{60}$
สูตรของ Haycock	$0.024265 \times W^{0.5378} \times H^{0.3964}$
สูตรของ Boyd	$0.0333 \times W^{(0.6157 - 0.0188 \log_{10} W)} \times H^{0.3}$

จงเขียนโปรแกรมที่รับค่าน้ำหนักและส่วนสูง แล้วแสดงค่าพื้นที่ผิวกายที่คำนวณได้จากสูตรทั้งสามข้างบนนี้

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนจริงแทนน้ำหนักหน่วยเป็นกิโลกรัม
บรรทัดที่สองเป็นจำนวนจริงแทนความสูงหน่วยเป็นเซนติเมตร

ข้อมูลส่งออก

ค่าพื้นที่ผิวกายที่คำนวณได้จากสูตรของ Mosteller, Haycock และ Boyd บรรทัดละค่า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
56 173	1.6404606399152375 1.6304868174022364 1.632155747802396
60 170	1.6832508230603465 1.680428314258862 1.6863370568707923
80.0 150.0	1.8257418583505538 1.8666576124395382 1.9007070607658065

Duration

โปรแกรมข้างล่างนี้รับเลขชั่วโมง นาที และวินาที ของเวลาเริ่มกับเวลาสิ้นสุด จากนั้นแสดงช่วงเวลาเป็นจำนวนชั่วโมง นาที และวินาที ระหว่างเวลาเริ่มถึงสิ้นสุด โดยมีข้อจำกัดว่า เวลาสิ้นสุดจะต้องไม่น้อยกว่าเวลาเริ่มต้น

```
h1 = int(input())
m1 = int(input())
s1 = int(input())
h2 = int(input())
m2 = int(input())
s2 = int(input())
t1 = h1*60*60 + m1*60 + s1
t2 = h2*60*60 + m2*60 + s2
dt = t2 - t1
dh = dt // (60*60)
dt -= dh * 60*60
dm = dt // 60
dt -= dm*60
ds = dt
print(str(dh)+":"+str(dm)+":"+str(ds))
```

เช่น ถ้าป้อนเลข **2 10 20 4 0 0** บรรทัดละจำนวน จะได้ผลลัพธ์คือ **1:49:40**

แต่ถ้าป้อน **2 0 0 1 0 0** บรรทัดละจำนวน จะได้ผลลัพธ์คือ **-1:0:0** ซึ่งผิด ที่ถูกควรเป็น **23:0:0**

จงปรับปรุง

จงปรับปรุงโปรแกรมข้างต้นให้ถูกต้องทั้งในกรณีที่รับเวลาสิ้นสุดมากกว่า น้อยกว่า หรือเท่ากับ เวลาเริ่มต้น (กำหนดให้ช่วงเวลาไม่เกิน 23:59:59)

ข้อแนะนำ : สมมติว่าเราสนใจเฉพาะเลขชั่วโมง การคำนวณช่วงเวลาจาก h1 ถึง h2

- แบบง่าย ๆ ก็เท่ากับ h2 - h1 เช่น h1 = 1 ถึง h2 = 2 ก็เท่ากับ h2 - h1 = 2 - 1 = 1 ชั่วโมง ซึ่งจะใช้ได้ก็เมื่อ h2 ≥ h1
- ถ้าสลับกัน ให้ h1 = 2 และ h2 = 1 ช่วงเวลา 2 นาฬิกา ถึง 1 นาฬิกา ย่อมไม่เท่ากับ 1 - 2 = -1 แต่เท่ากับ 23 ชั่วโมง ถ้าดูดี ๆ 23 = 24 + (-1) จึงขอแก้สูตรช่วงเวลาจาก h1 ถึง h2 ให้เท่ากับ 24 + (h2 - h1) ก็จะใช้ได้ในกรณี h2 < h1
- ถ้าปรับสูตรให้เป็น (24 + (h2 - h1)) % 24 ก็สามารถใช้ได้ไม่ว่า h2 ≥ h1 หรือ h2 < h1 (ลองดูเอง)
- (หรือใช้สูตรแค่ (h2 - h1) % 24 ก็ใช้ได้เหมือนกัน จะเข้าใจตรงนี้ ต้องเข้าใจการใช้ % กับจำนวนลบ ซึ่งไม่ขออธิบาย)
- จากแนวทางข้างบนนี้ สามารถนำไปปรับให้ใช้กับการคำนวณช่วงเวลาเมื่อกำหนดเป็นชั่วโมง นาที และวินาที ตามโจทย์

ข้อมูลนำเข้า

สามบรรทัดแรกรับ เลขชั่วโมง นาที และวินาที ของเวลาเริ่มต้น บรรทัดละจำนวน

ตามด้วยอีกสามบรรทัดที่รับ เลขชั่วโมง นาที และวินาที ของเวลาสิ้นสุด บรรทัดละจำนวน

(ชั่วโมงเป็นจำนวนเต็ม 0 ถึง 23 ส่วนนาทีและวินาทีเป็นจำนวนเต็ม 0 ถึง 59)

ข้อมูลส่งออก

ช่วงเวลาตั้งแต่เวลาเริ่มจนสิ้นสุด (ที่รับเข้ามา) แสดงเป็นจำนวนชั่วโมง นาที และวินาที ในรูปแบบที่แสดงในตัวอย่าง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2 10 20 4 0 0	1:49:40
18 10 10 19 0 0	0:49:50
19 0 0 18 10 10	23:10:10

Extract Functions

โปรแกรมข้างล่างนี้รับน้ำหนักและความสูงทางแป้นพิมพ์ เพื่อคำนวณและแสดงค่าพื้นที่ผิวกาย 3 แบบ ตามสูตรทางขวานี้

สูตร Mosteller	$\frac{\sqrt{W \times H}}{60}$
สูตร Du Bois	$0.007184 \times W^{0.425} \times H^{0.725}$
สูตร Fujimoto	$0.008883 \times W^{0.444} \times H^{0.663}$

```
w = float(input()) # body weight
h = float(input()) # body height
mosteller = ((w*h)**0.5) / 60
du_bois = 0.007184 * (w**0.425) * (h**0.725)
fujimoto = 0.008883 * (w**0.444) * (h**0.663)
print("Mosteller =", mosteller)
print("Du Bois =", du_bois)
print("Fujimoto =", fujimoto)
```

จงปรับโปรแกรมข้างบนนี้ใหม่ตามโครงของโปรแกรมข้างล่างนี้ ซึ่งแยกการคำนวณแต่ละสูตรเป็นฟังก์ชัน 3 ฟังก์ชัน และเพิ่มฟังก์ชัน **main** ที่ทำหน้ารับน้ำหนัก ความสูง เรียกใช้ฟังก์ชันทั้งสาม และแสดงผลลัพธ์ เพื่อให้ทำงานเหมือนเดิม

```
def mosteller(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Mosteller formula
    ???

def du_bois(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Du Bois formula
    ???

def fujimoto(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Fujimoto formula
    ???

def main():
    weight = float(input())
    height = float(input())
    ???
    ???
    ???
    print("Mosteller =", round(???, 5))
    print("Du Bois =", round(???, 5))
    print("Fujimoto =", round(???,5))

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(mosteller(56,173))`	`1.6404606399152375`
`print(du_bois(56,173))`	`1.6669772003009131`
`print(fujimoto(56,173))`	`1.6165149017101`

`main()`
`56`
`173`
	`Mosteller = 1.64046`
	`Du Bois = 1.66698`
	`Fujimoto = 1.61651`

การหารากที่สามด้วยเครื่องคิดเลขแบบธรรมดา

จากความรู้พื้นฐานเกี่ยวกับอนุกรมเรขาคณิตที่ว่า $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$ เมื่อ $|x| < 1$ ถ้าให้ $x = \frac{1}{4}$ จะได้ว่า

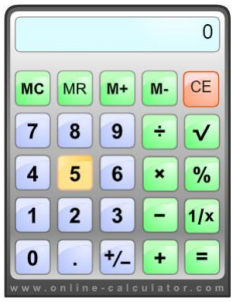
$$\frac{1}{1-1/4} - 1 = \frac{1}{3} = \frac{1}{4^1} + \frac{1}{4^2} + \frac{1}{4^3} + \frac{1}{4^4} + \frac{1}{4^5} + \frac{1}{4^6} + \frac{1}{4^7} + \frac{1}{4^8} + \ldots \quad \text{(สมการที่ 1)}$$

$$= \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^{10}} + \frac{1}{2^{12}} + \frac{1}{2^{14}} + \frac{1}{2^{16}} + \ldots \quad \text{(สมการที่ 2)}$$

$$= \frac{1}{2^2}\left(1 + \frac{1}{2^2}\right)\left(1 + \frac{1}{2^4}\right)\left(1 + \frac{1}{2^8}\right) \ldots \quad \text{(สมการที่ 3)}$$

ใช้สมการที่ 2 หารากที่สามของ y จะได้ $y^{\frac{1}{3}} = y^{\frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^{10}} + \frac{1}{2^{12}} + \frac{1}{2^{14}} + \frac{1}{2^{16}} + \ldots}$ (สมการที่ 4)

ใช้สมการที่ 3 หารากที่สามของ y จะได้ $y^{\frac{1}{3}} = y^{\frac{1}{2^2}(1 + \frac{1}{2^2})(1 + \frac{1}{2^4})(1 + \frac{1}{2^8}) \ldots}$ (สมการที่ 5)



จากผลที่ได้ แสดงว่า เราสามารถหารากที่สาม ด้วยเครื่องคิดเลขแบบธรรมดา (ที่มีปุ่ม $\sqrt{x}$ แต่ไม่มีปุ่ม x^y) เช่น $8^{(1/2^2+1/2^4)}$ หาค่าได้ด้วยการกดปุ่มของเครื่องคิดเลขตามลำดับดังนี้ 8 √ √ × 8 √ √ √ √ =

หรือ $8^{(1/2^2)(1+1/2^2)(1+1/2^4)}$ หาค่าได้ด้วยการกดปุ่มตามลำดับดังนี้ 8 √ √ × √ √ × √ √ √ √ =

ตารางข้างล่างนี้ แสดงการใช้สมการที่ 4 กับ 5 เพื่อคำนวณรากที่สามของ 27 ด้วยโปรแกรม Calculator ของ Windows พบว่า การใช้สมการที่ 5 ใช้จำนวนครั้งของการกดปุ่มที่น้อยกว่าการใช้สมการที่ 4 (โดยได้ผลที่มีความแม่นยำพอ ๆ กัน)

ลำดับปุ่มที่กด เพื่อคำนวณตามสมการที่ 4	ลำดับปุ่มที่กด เพื่อคำนวณตามสมการที่ 5
CE MC 2 7 M+ √ 2 ครั้ง × MR √ 4 ครั้ง × MR √ 6 ครั้ง × MR √ 8 ครั้ง × MR √ 10 ครั้ง × MR √ 12 ครั้ง × MR √ 14 ครั้ง × MR √ 16 ครั้ง = ได้คำตอบ 2.999949709941997 (กดปุ่มทั้งสิ้น 92 ครั้ง)	CE 2 7 √ 2 ครั้ง × √ 2 ครั้ง × √ 4 ครั้ง × √ 8 ครั้ง = ได้คำตอบ 2.999949709941997 (กดปุ่มทั้งสิ้น 23 ครั้ง)

ถ้าต้องการคำตอบที่แม่นยำขึ้นก็สามารถกดคำนวณต่อได้ ในตัวอย่างที่แสดงนี้ ขอกดเพื่อให้การคำนวณด้วยสมการ 4 และ 5 ได้ผลลัพธ์ที่มีความแม่นยำพอ ๆ กัน

จงเขียนฟังก์ชันต่าง ๆ ในโปรแกรมข้างล่างนี้

```
def sqrt_n_times(x, n):
    # คืนค่าที่เสมือนการนำค่าใน x มากดปุ่ม √ เป็นจำนวน n ครั้ง

    ???

def cube_root(y):
    # คืนค่าประมาณของรากที่สามของ y โดยใช้วิธีที่เสมือนการกดปุ่มด้วยสูตร
    # y^(1/2^2)(1+1/2^2)(1+1/2^4)(1+1/2^8)(1+1/2^16)(1+1/2^32)
    # ข้อแนะนำ: เรียกใช้ฟังก์ชัน sqrt_n_times

    ???

def main():
    q = float(input())
    print(cube_root(q))

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(sqrt_n_times(10**8,3))`	`10.0`
`print(round(cube_root(27), 4))`	`3.0`
`print(cube_root(5)**3, (5**(1/3))**3)`	`5.000000000000001 4.999999999999998`
`main()` `27`	`2.999999999999999`

ช่วงเวลา

จงเขียนฟังก์ชันต่าง ๆ ให้ทำงานตามที่เขียนใน comment ของโปรแกรมข้างล่างนี้ (สองฟังก์ชันแรกทำงานถูกต้องแล้ว ไม่ต้องเขียน)

```
def str2hms(hms_str):
    # คืนจำนวนชั่วโมง นาที และวินาที ที่ดึงมาจากสตริง hms
    # เช่น str2hms("10:03:29") ได้ 10,3,29
    t = hms_str.split(':')
    return int(t[0]),int(t[1]),int(t[2])

def hms2str(h,m,s):
    # คืนสตริงในรูปแบบ HH:MM:SS ที่นำจำนวนชั่วโมง นาที และวินาทีมาจาก h,m และ s
    # เช่น hms2str(10,3,29) ได้ "10:03:29"
    return ('0'+str(h))[-2:] + ':' + \
           ('0'+str(m))[-2:] + ':' + \
           ('0'+str(s))[-2:]

def to_sec(h,m,s):
    # คืนจำนวนวินาทีทั้งหมดนับจากเที่ยงคืนจากถึงเวลา h:m:s
    # เช่น to_sec(10,3,29) ได้ 36209

    ???

def to_hms(s):
    # คืนจำนวนชั่วโมง นาที และวินาที ที่หามาจากจำนวนวินาที่ s ทั้งหมดนับจากเที่ยงคืน
    # เช่น to_hms(36209) ได้ 10,3,29

    ???

def diff(h1,m1,s1,h2,m2,s2):
    # คืนจำนวนชั่วโมง นาที และวินาที่ จะเป็นช่วงเวลาตั้งแต่เวลา h1,m1,s1 จนถึง h2,m2,s2
    # เช่น diff(10,57,57, 12,0,0) ได้ 1,2,3
    # หมายเหตุ เวลา h1,m1,s1 ที่ได้รับ ไม่มากกว่า h2,m2,s2 แน่ ๆ
    # (เช่น ไม่มีกรณีให้หาช่วงเวลาตั้งแต่ 23,50,50 ถึง 2,1,1 แน่ ๆ)

    ???

def main():
    # ฟังก์ชันนี้รับเวลาเริ่มต้น และเวลาสิ้นสุด ในรูปแบบ HH:MM:SS
    # เพื่อแสดงช่วงเวลาตั้งแต่เริ่มจนถึงสิ้นสุด ในรูปแบบ HH:MM:SS
    # ดูตัวอย่างในตารางข้างล่าง
    hms_start = input()
    hms_end = input()

    ???

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(to_sec(10,3,29))`	`36209`
`h,m,s = to_hms(36209); print(h,m,s)`	`10 3 29`
`h,m,s = to_hms(36209); print(hms2str(h,m,s))`	`10:03:29`
`dh,dm,ds = diff(10,57,57,12,0,0); print(dh,dm,ds)`	`1 2 3`

`main()`
`10:57:57`
`12:00:00`
	`01:02:03`