```python
get_triangle_area(input(), input())  # aufgabe 1a

# aufgabe 1b
'''
Eine siwtch-case Anweisung kann man mit if-elif Anweisungen simulieren, indem man die erste if-Anweisung statt switch-Anweisung nutzt und
die weiteren elif-Anweisungen statt case, das könnte so aussehen:
input(letter)
if (letter == 'a'): $Anweisung 1$
elif (letter == 'b'): $Anweisung 2$
elif (letter == 'c'): $Anweisung 3$
...
elif (letter == 'y'): $Anweisung 27$
else: $Anweisung 28$
'''

# aufgabe 1c
'''
Bei einer while-Schleife muss der Zähler jedes Mal innerhalb der Schleife geändert werden, also die Schleife:
for i in range(a, b, c) könnte man durch die Schleife ersetzen:
    i = a
    while i < b:
        ...
    i += c
'''

print(check(fill()))  # aufgabe 1d
print(average(fill_until_zero()))  # aufgabe 1e
print(get_price(int(input())))  # aufgabe 2a
```

```python
# aufgabe 2b
'''
if $Bedingung$ then $Anweisung1$
if $nicht Bedingung$ then $Anweisung2$

Mit einem konkreten Beispiel:
a = input()
if a % 2 == 0 then print("gerade")
if a % 2 != 0 then print("ungerade")

Das heißt man muss die zweite Bedingung (oder die Gegenbedingung) auch ausdrücklich schreiben. Der Nachteil ist aber, dass
dabei Vergleiche gemacht werden, die tatsächlich nicht nötig sind.
'''

# aufgabe 2c
'''
Bei einer do-while Schleife wird eine Answeisung ausgeführt solange die nachfolgende Bedingung richtig ist.
Da die Bedingung nachfolgend ist, wird die Anweisung mindestens einmal ausgeführt und erst dann wird überprüft, ob
die Bedingung richtig ist.
Beispiel in Python:
$Anweisung$
while $Bedingung$:
    $Anweisung$

input(a)
a *= 2
while a <= 100: a *=2
'''
```

```python
print(get_variability(fill()))  # aufgabe 2d
print_pyramid(int(input()))  # aufgabe 2e
simulate_coin_experiment( n: 50,  n_exp: 10000)  # aufgabe 3
```

```python
import random


1 usage
def get_triangle_area(a, ha):
    return a * ha/2


2 usages
def fill():
    arr: list[int] = list()
    for i in range(3):
        arr.append(int(input()))
    return arr


1 usage
def fill_until_zero():
    arr: list[int] = list()
    while True:
        k = int(input())
        if k != 0:
            arr.append(k)
        else:
            break
    return arr
```

```python
1 usage
def average(arr):
    return sum(arr) / len(arr)


1 usage
def check(arr):
    for i in arr:
        if arr.count(i) != 1: return False
    return True


1 usage
def get_variability(arr):
    return len(set(arr))


1 usage
def get_price(age: int):
    if age < 12:
        return 10
    elif age < 18:
        return 12
    elif age < 65:
        return 14
    else:
        return 12
```

```
1 usage
def print_pyramid(n):
    for i in range(1, n + 1):
        k = i
        print_spaces(n - i)
        while k >= 1:
            print(k, sep='', end='')
            k -= 1
        k = k + 2
        while k <= i:
            print(k, sep='', end='')
            k += 1
        print()


1 usage
def simulate_coin_experiment(n, n_exp):
    total = 0
    for i in range(n_exp):
        z = 0
        counter = 0
        while z < n:
            if random.random() > 0.5 or z == 0: z += 1
            else: z -= 1
            counter += 1
        total += counter
    print(total/n_exp)
```