Numpy

ข้างล่างนี้คือฟังก์ชันที่ทำงานถูกต้อง โดยใช้วงวน **for** ช่วยในการประมวลผล

download code นี้

```
import numpy as np

def f1(M, c):
    # M: a 2-D numpy array
    # c: float
    A = np.zeros_like(M)
    for i in range(M.shape[0]):
        for j in range(M.shape[1]):
            A[i,j] = M[i,j]*c
    return A

def f2(U, V):
    # U and V: 1-D numpy arrays of equal size
    d = 0
    for i in range(len(U)):
        d += U[i]*V[i]
    return d

def f3(M):
    # M: a 2-D numpy array
    t = np.ndarray(M.shape[::-1])
    for i in range(M.shape[1]):
        for j in range(M.shape[0]):
            t[i,j] = M[j,i]
    return t

def f4(x, y, dx, dy, k, R):
    # x, y, dx, dy: 1-D numpy arrays of all equal size
    # k, R: float

    neighbors = [(x[i]-x[k])**2 + (y[i]-y[k])**2 <= R**2
                 for i in range(len(x))]
    sx = sy = 0
    for i in range(len(neighbors)):
        # True is equal to 1 and False is equal to 0
        sx += neighbors[i] * dx[i]
        sy += neighbors[i] * dy[i]
    t = np.arctan2(sy, sx)
    return np.cos(t), np.sin(t)

#----- DON'T modify any of the following lines -----
for k in range(int(input())):
    exec(input().strip())
```

จงเขียนการทำงานในฟังก์ชัน **f1**, **f2**, **f3** และ **f4** ข้างบนนี้ใหม่ โดย

- **ห้ามใช้คำสั่ง `for, while, recursive, itertools, comprehension, map, reduce`**
- **จะตรวจเงื่อนไขข้างบนนี้หลังสอบเสร็จ หากผิดเงื่อนไข ได้ศูนย์**

ถ้าใช้วงวน ผลลัพธ์ของการตรวจในเกรดเดอร์จะแสดง **x**

และจะตรวจหลังสอบอีกครั้ง

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2 a = np.arange(0,10,0.5).reshape((4,5)) print(f1(a,2.0))	[[0. 1. 2. 3. 4.] [5. 6. 7. 8. 9.] [10. 11. 12. 13. 14.] [15. 16. 17. 18. 19.]]
2 a = np.array([1,2,3,2,1], float) print(f2(a,a))	19.0
1 print(f3(np.arange(10,20,1.0).reshape((2,5))))	[[10. 15.] [11. 16.] [12. 17.] [13. 18.] [14. 19.]]
6 x = np.array([5, 10, 15, 20], float) y = np.array([10, 5, 10, 5], float) dx = np.array([0, 0.8, 0, -1]) dy = np.array([1, 0.5, -1, 0]) t = f4(x, y, dx, dy, 1, 20) print(round(t[0],4), round(t[1],4))	-0.3714 0.9285