2559_2_NumPy_L4

จงเขียนฟังก์ชันต่าง ๆ ที่เว้นว่างในโปรแกรมข้างล่างนี้ ที่มีข้อกำหนดของพารามิเตอร์ และผลลัพธ์ที่ได้ตามตารางนี้

function	return value

checker(n)
 	
n เป็นจำนวนเต็ม
	array สองมิติขนาด **n × n** เก็บค่า **0** กับ **1** สลับตำแหน่งคล้าย ๆ ตารางหมากฮอส เช่น **print(checker(5))** ได้
	`[[0 1 0 1 0]`
	` [1 0 1 0 1]`
	` [0 1 0 1 0]`
	` [1 0 1 0 1]`
	` [0 1 0 1 0]]`

collide(e,C)
 	
e เก็บวงกลม **1** วง
C เก็บวงกลมหลายวง
e เป็นอาเรย์หนึ่งมิติ **3** ช่องเก็บพิกัด **x,y** และ รัศมี
C เป็นอาเรย์สองมิติ หนึ่งแถวเก็บวงกลม **1** วง
C จึงเก็บวงกลมจำนวน **C.shape[0]** วง
	คืนอาเรย์สองมิติ เก็บวงกลมใน **C** เฉพาะวงทับหรือแตะกับวงกลม **e** เช่น
	`e = np.array([5,0,3])`
	`C = np.array([[0,0,1],[0,0,2],[0,0,3]])`
	`print(collide(e,C))`
	
	ได้
	
	`[[0 0 2]`
	` [0 0 3]]`

matrix_chain_mult(M,order) **M** เก็บอาเรย์ของเมทริกซ์ เมทริกซ์เป็นอาเรย์สองมิติ ดังนั้น **M** จึงเป็นอาเรย์สามมิติ **M[0],M[1],...** เป็นเมทริกซ์ที่สามารถนำมาคูณกันได้ตามลำดับ **M[0]×M[1]×M[2]×...** แต่การคูณเมทริกซ์ทั้งหมดก็ไม่จำเป็นต้องทำตามลำดับจากซ้ายไปขวา เช่น **M[0]×M[1]×M[2]** จะคูณตามลำดับแบบ **(M[0]×M[1])×M[2]** หรือ แบบ **M[0]×(M[1]×M[2])** ก็ได้เหมือนกัน **order** คือ **list** ของ **index** ของเมทริกซ์ใน **M** ที่นำมาคูณ เช่น **order** ของ **(M[0]×M[1])×M[2]** คือ **0,1,2** ขณะที่ของ **M[0]×(M[1]×M[2])** คือ **1,2,0** ให้ฟังก์ชันนี้คูณเมทริกซ์ใน **M** ตามลำดับที่กำหนดใน **order**	ผลคูณของเมทริกซ์ใน **M** ตามลำดับที่กำหนดใน **order** คำเตือน ถึงแม้ว่าผลคูณของ **(M[0]×M[1])×M[2]** กับ **M[0]×(M[1]×M[2])** เหมือนกัน แต่เวลาและหน่วยความจำที่ใช้ในการคูณจะไม่เหมือนกัน

```
import numpy as np

def checker(n):
    _____________________________________  # หาวิธีเขียนอย่างมากสามคำสั่ง

def collide(e,c):
    _____________________________________  # หาวิธีเขียนอย่างมากสองคำสั่ง

def matrix_chain_mult(M, order):
    _____________________________________  # อย่าลืมคูณตามลำดับที่กำหนดใน order


exec(input().strip())  # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

Input / Output

Input:
`print(checker(10))`
 	
Output:
`[[0 1 0 1 0 1 0 1 0 1]`
` [1 0 1 0 1 0 1 0 1 0]`
` [0 1 0 1 0 1 0 1 0 1]`
` [1 0 1 0 1 0 1 0 1 0]`
` [0 1 0 1 0 1 0 1 0 1]`
` [1 0 1 0 1 0 1 0 1 0]`
` [0 1 0 1 0 1 0 1 0 1]`
` [1 0 1 0 1 0 1 0 1 0]`
` [0 1 0 1 0 1 0 1 0 1]`
` [1 0 1 0 1 0 1 0 1 0]]`

Input:
`print(collide(np.array([3,3,1]),np.array([[1,1,1],[2,2,2],[2,3,1],[4,4,2],[3,-3,3]])))`
 	
Output:
`[[2 2 2]`
` [2 3 1]`
` [4 4 2]]`

Input:
`a=np.array([[1,2,3],[4,1,2]]);print(matrix_chain_mult(np.array([a,a.T,a,a.T]),[1,2,0,3]))`
 	
Output:
`[[340 420]`
` [420 585]]`
 	
(M[0] × (M[1] × M[2])) × M[3]