First Fit & Best Fit

กำหนดให้มีรายการของจำนวนเต็ม (แต่ละจำนวนมีค่าได้ตั้งแต่ 1 ถึง 100) เช่น [10, 20, 90, 50, 10, 20] คำถามที่น่าสนใจคือ จะแบ่งรายการนี้ออกเป็นรายการย่อย ๆ อย่างไร ที่ทำให้แต่ละรายการย่อยมีผลรวมของจำนวนเต็มไม่เกิน 100 และได้จำนวนรายการย่อยที่น้อยสุด เช่น [10, 20, 90, 50, 10, 10] แบ่งได้ดีสุด คือ [[10, 20, 50, 10], [90, 10]]

ขอนำเสนอวิธีแบ่งแบบง่าย (ที่ไม่ได้จำนวนรายการที่น้อยสุด) โดยพิจารณาข้อมูลทีละตัว แล้วเลือกใส่ในรายการย่อยที่มีอยู่ โดยมีวิธีการเลือกรายการย่อย 2 วิธี

- First Fit วิธีนี้หารายการย่อย (จากซ้ายไปขวา) พบอันที่ใส่ข้อมูลใหม่ได้ ก็ใส่เลย เช่น ต้องการใส่ 20 ลงใน [[90, 5], [50], [70,10]] พบว่า 20 ใส่ใน [90, 5] ไม่ได้ แต่ใส่ใน [50] ได้ ก็ใส่เลย เป็น [[90, 5], [50, 20], [70,10]]
- Best Fit วิธีนี้พิจารณาทุกรายการย่อยที่ใส่ข้อมูลใหม่ได้ แล้วเลือกใส่รายการที่จะทำให้ผลรวมใกล้ 100 ที่สุด เช่น ต้องการใส่ค่า 20 ลงใน [[90, 5], [50], [70,10]] พบว่าใส่ 20 ใน [90, 5] ไม่ได้ แต่ใส่ใน [50] กับ [70,10] ได้ จะเลือกใส่ใน [70,10] เพราะได้ผลที่ใกล้ค่า 100 ที่สุด ได้ผลเป็น [[90, 5], [50], [70,10,20]]

ในกรณีที่ ไม่สามารถหารายการย่อยใดเลยที่ใส่ข้อมูลใหม่ได้ (เพราะใส่แล้วเกินร้อย) ก็สร้างรายการย่อยใหม่ต่อท้ายของที่มีอยู่

จงเขียนสี่ฟังก์ชัน ที่ทำงานตาม comment ทีเขียนข้างล่างนี้

```
def first_fit(L, e):      # นำ e ใส่รายการย่อยใน L ด้วยวิธี first fit

def best_fit(L, e):       # นำ e ใส่รายการย่อยใน L ด้วยวิธี best fit

def partition_FF(D):      # คืนลิสต์ของลิสต์ที่ได้จากการแบ่งข้อมูลใน D ด้วย first fit

def partition_BF(D):      # คืนลิสต์ของลิสต์ที่ได้จากการแบ่งข้อมูลใน D ด้วย best fit

exec(input().strip())     # ต้องมีคำสั่งนี้ ตรงนี้ ตอนส่งให้ Grader ตรวจ
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`L=[[50],[90]];first_fit(L,10);print(L)`	`[[50, 10], [90]]`
`L=[[50],[90]];best_fit(L,10);print(L)`	`[[50], [90, 10]]`
`print(partition_FF([50,90,10,80,50,20]))`	`[[50, 10, 20], [90], [80], [50]]`
`print(partition_BF([50,90,10,80,50,20]))`	`[[50, 50], [90, 10], [80, 20]]`