กระแสเงินสด

กำหนดให้เราแทนเงินด้วย dict ที่มี key เป็นมูลค่าของเหรียญหรือธนบัตร และ value คือจำนวนเหรียญหรือธนบัตรที่มี
เช่น **{100: 5, 50: 2, 10: 5, 1: 15}** แทนธนบัตรหนึ่งร้อยบาท 5 ใบ, ห้าสิบ 2 ใบ, สิบบาท 5 ใบ และเหรียญบาท 15 เหรียญ

จงเขียนฟังก์ชันต่าง ๆ ข้างล่างนี้ (ดูตัวอย่างข้างล่างประกอบ)

- **total(pocket)** คืนจำนวนเงินรวมใน **pocket**
- **take(pocket, money)** เติมเงินจาก **money** เข้าใน **pocket** (ทั้งคู่เป็น dict ที่เก็บเงิน)
- **pay(pocket, amt)** ตัดเงินใน **pocket** เป็นจำนวน **amt** (เป็นจำนวนเต็ม) ตัวฟังก์ชันจะคืนเงิน (เป็น dict) ที่จ่ายออกไป ในกรณีที่จ่ายเป็นจำนวน **amt** ไม่ได้ ให้คืน dict ว่าง
  โดยการจ่ายเงิน ใช้วิธีการเลือกธนบัตรหรือเหรียญที่มีในกระเป๋าที่มีมูลค่าสูงสุดที่จ่ายได้ก่อน เช่น {100: 5, 50: 2, 10:5, 1: 15} ต้องการจ่ายออก 57 ก็จะหยิบ 50 ออกหนึ่งใบ และ 1 ออกเจ็ดเหรียญ

```
def total(pocket):


def take(pocket, money_in):


def pay(pocket, amt):


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

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`p={100:2, 50:2, 5:2, 1:2};print(total(p))`	`312`
`p={100:5};take(p,{100:2, 1:3});print(p)`	`{100: 7, 1: 3}`
`p={100:5};take(p,{100:0, 1:0});print(p)`	`{100: 5, 1: 0}`
`p={10:5, 1:7};print(pay(p, 12));print(p)`	`{10: 1, 1: 2}` `{10: 4, 1: 5}`
`p={10:5, 1:7};print(pay(p, 18));print(p)`	`{}` `{10: 5, 1: 7}` จ่ายไม่ได้
`p={10:5, 1:7};print(pay(p, 100));print(p)`	`{}` `{10: 5, 1: 7}` จ่ายไม่ได้
`p={10:5, 1:7};print(pay(p, 57));print(p)`	`{10: 5, 1: 7}` `{10: 0, 1: 0}` จ่ายหมดกระเป๋า