ปัญหา อารีน่าของ SU 2 (AoSU2) [8 คะแนน]

[ทรัพยากรการคำนวณ: 1 วินาที, 32 MB]

เนื่องจากธรรมชาติของเกมแบบ Online Battle Arena มักจะเน้นที่ระบบผู้เล่นหลายคน (Multiplayer) ทำให้ได้ชื่อเรียกแบบเต็มเป็น Multiplayer Online Battle Arena หรือที่เรานิยมเรียกสั้น ๆ ว่า MOBA ปัจจัยสำคัญหนึ่งในการเล่นเพื่อชนะก็คือความสมดุลของทีม ซึ่งเกิดจากการเลือกฮีโร่ให้มีความหลากหลายสนับสนุนกันได้ดี รวมถึงทำให้อีกฝ่ายแก้เกมได้ยากขึ้นด้วย

ความหลากหลายของฮีโร่นี้เกิดขึ้นมาเพราะฮีโร่นั้นแท้จริงมีหลายชนิด แต่ก่อนที่เราจะไปจัดการประเด็นเรื่องความสมดุลของทีม เราจะจัดการเรื่องทีมที่ถูกกฎให้ได้ก่อน ซึ่งทีมที่ถูกกฎจะมีสองปัจจัยคือ ก. มีฮีโร่ที่เป็นสมาชิกของทีม 5 คนพอดี และ ข. ฮีโร่ไม่ซ้ำกัน (ถ้าชื่อฮีโร่เหมือนกันถือว่าซ้ำกัน)

ในปัญหานี้ เราจะยังไม่แยกชนิดฮีโร่ และเน้นไปเฉพาะเรื่องทีมที่ถูกกฎก่อน นั่นคือเราจะสร้างคลาส Team ที่จัดการทีมผู้เล่นขนาด 5 คนโดยมีเมธอดสำคัญที่เราจะเขียนขึ้นมาดังนี้

1. ตัวสร้าง ซึ่งจะรับอาเรย์มาหนึ่งตัว โดยอาเรย์ตัวนี้เก็บข้อมูลการเลือกฮีโร่ของทีมเอาไว้ และตัวสร้างจะตรวจสอบความถูกต้องของการเลือกฮีโร่ ซึ่งถ้าไม่พบข้อผิดพลาดใด ๆ ก็จะคัดลอกข้อมูลภายในอาเรย์มาเก็บไว้ในตัวแปรสมาชิกคลาสชื่อ members ซึ่งแทนการเลือกฮีโร่ที่เป็นสมาชิกในทีมของผู้เล่น ส่วนถ้ามีข้อผิดพลาดก็จะเซ็ตให้ members มีค่าเป็น null

2. เมธอด is5MemberTeam ซึ่งจะรับพารามิเตอร์แบบเดียวกับตัวสร้าง เพื่อตรวจสอบว่ามีการเลือกฮีโร่ครบ 5 คนพอดี ซึ่งจะรู้ได้หากตรวจสอบความยาวอาเรย์ และที่สำคัญคือจะต้องระวังกรณีที่พารามิเตอร์เป็นค่า null ด้วย ซึ่งเกิดจากการที่ยังไม่มีการเลือกฮีโร่เลย

ในกรณีที่ทีมยังมีสมาชิกไม่ครบ เมธอดจะขึ้นคำเตือนว่า **member is missing** และคืนค่า **false** หากมีสมาชิก 5 คนพอดี เมธอดจะขึ้นข้อความว่า **full team** และคืนค่า **true** แต่หากมีสมาชิกเกิน 5 คน เมธอดจะขึ้นคำเตือนว่า **too many members** และคืนค่า **false**

3. เมธอด **isValid** ทำหน้าที่ตรวจสอบได้ว่าฮีโร่ในทีมซ้ำหรือไม่ ซึ่งรับพารามิเตอร์แบบเดียวกันกับตัวสร้างเช่นกัน หากมีฮีโร่ที่ชื่อซ้ำกัน แสดงว่ามีการซ้ำ เมธอดจะขึ้นข้อความเตือนว่า **cannot select same hero:** ตามด้วยชื่อฮีโร่ที่ซ้ำที่พบเป็นตัวแรก และคืนค่า **false** ไม่เช่นนั้นก็จะขึ้นข้อความว่า **valid hero selection** และคืนค่า **true**

อย่างไรก็ตาม ในตอนต้นเมธอดนี้ก็จะตรวจจำนวนสมาชิกในทีมก่อนในลักษณะเดียวกันกับ **is5MemberTeam** ซึ่งถ้าไม่ได้มีสมาชิก **5** คนพอดี เมธอด **isValid** จะคืนค่า **false** กลับไปโดยไม่พิมพ์ข้อความใด ๆ ออกมา

การทำงานของตัวสร้าง

วกกลับมาที่การทำงานของตัวสร้าง ตัวสร้างของเราจะเริ่มจากการตรวจสอบว่าสมาชิกมี 5 ตัวพอดีหรือไม่ โดยการเรียกใช้ **is5MemberTeam** ซึ่งถ้าไม่ใช่ ตัวสร้างก็จะหยุดการทำงานและ **return** กลับไปทันที แต่ถ้าใช่ จะต้องตรวจสอบการซ้ำซ้อนในการเลือกฮีโร่ด้วยการเรียกใช้ **isValid** ซึ่งถ้าไม่มีปัญหาเรื่องการซ้ำ ตัวสร้างก็ถือว่าทีมถูกกฎและจะบันทึกข้อมูลทีมลง **members** ด้วยวิธีที่ชี้แจงไว้ก่อนหน้า ไม่เช่นนั้นก็จะ **return** โดยไม่บันทึกข้อมูล **members**

จงสร้างคลาส **Team** ตามข้อกำหนดดังกล่าว และใช้เมธอด **main** ที่กำหนดให้และห้ามทำการเปลี่ยนแปลงใด ๆ ในคลาส **AoSU2** ยกเว้นส่วนที่มีการเว้นไว้สำหรับการเติมโค้ดโดยเฉพาะ ถ้าหากมีการแก้ไขใด ๆ ในคลาส **AoSU2** นอกเหนือไปจากนั้นและมีผล

ต่อคำตอบแล้วได้คะแนน คะแนนดังกล่าวจะกลายเป็นแต้มลบ เช่น ถ้าได้คะแนน 60 ก็จะถูกปรับเป็น -60 แทน

```
import java.util.Scanner;

class Team {

  // Enter your code here.

}

public class AoSU2 {
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int N = scan.nextInt();
    ________ members ________;
    if(________) {
      members = new ________;
      for(int i = 0; i < N; ++i) {
        String name = scan.next();
        int hp = scan.nextInt();
        int attkType = scan.nextInt();
        int attkDmg = scan.nextInt();
        int phyDef = scan.nextInt();
        int magDef = scan.nextInt();

        ________ = new Hero(name, hp, attkType,
                                attkDmg, phyDef, magDef);
      }
    }
    Team t = new Team(members);
    System.out.println(t.is5MemberTeam(members));
    System.out.println(t.isValid(members));
  }
}
```

ข้อมูลเข้า

บรรทัดแรก	เป็นค่า N ซึ่งคือจำนวนฮีโร่ที่จะอยู่ในทีม โดยที่ 0 <= N <= 20
บรรทัดที่สอง	เป็นข้อมูลฮีโร่ตัวแรก ซึ่งอยู่ในลักษณะเดียวกันกับข้อ AoSU1 แต่ถูกจัดใหม่ให้บรรทัดทั้ง 4 ถูกยุบมาเป็นบรรทัดเดียว

อีก N-1 บรรทัด	เป็นข้อมูลฮีโร่ที่เหลือ หนึ่งตัวต่อบรรทัด

ผลลัพธ์

เป็นสิ่งที่ตัวสร้างพิมพ์ออกมา ตามด้วยสิ่งที่พิมพ์และค่าที่คืนจาก `is5MemberTeam` และสิ่งที่พิมพ์ออกมาและค่าที่คืนจาก `isValid`

ตัวอย่าง

ข้อมูลเข้า	ผลลัพธ์
0	member is missing member is missing false false
3 Van 100 1 70 50 20 Taara 100 1 -70 50 20 Alice 100 2 70 50 20	warning: value cannot be negative member is missing member is missing false false
3 Van 100 1 70 50 20 Van 100 1 -70 50 20 Van 100 2 70 -50 20	warning: value cannot be negative warning: value cannot be negative member is missing member is missing false false
5 Van 1000 1 70 50 20 Taara 1000 1 -70 50 20 Alice 1000 2 70 50 20 Maloch 1000 1 70 50 20 Zuka 800 1 90 40 40	warning: value cannot be negative full team valid hero selection full team true valid hero selection true
5 Van 1000 1 70 50 20 Taara 1000 1 -70 50 20 Alice 1000 2 70 50 20 Maloch 1000 1 70 50 20 Alice 800 1 90 40 40	warning: value cannot be negative full team cannot select same hero: Alice full team true cannot select same hero: Alice false

6 Zuka 1000 1 70 50 20 Taara 1000 1 -70 50 20 Alice 1000 2 70 50 20 Maloch 1000 1 70 50 20 Zuka 800 1 90 40 40 Violet 1000 1 70 50 20	warning: value cannot be negative too many members too many members false false

เงื่อนไขการให้คะแนน: โปรแกรมจะต้องทำงานถูกต้องอย่างน้อย **60%** ของชุดข้อมูลทดสอบจึงจะได้คะแนน

คำแนะนำ: การจะตรวจหาว่ามีชื่อซ้ำหรือไม่ เราจะเริ่มจากการเทียบชื่อของคนแรก กับคนอื่น ๆ ที่เหลือ (ซึ่งก็คือคนที่สองขึ้นไป) และต่อด้วยการเทียบชื่อคนที่สองกับคนที่สามขึ้นไป (หรือจะใช้การเทียบกับคนอื่น ๆ ที่เหลือทั้งหมดก็ยังได้ แต่โปรแกรมจะช้าลงเล็กน้อย)