**เฉลยแบบทดสอบย่อย**

**วิชา: ENGCC304 การเขียนโปรแกรมคอมพิวเตอร์**

**ตอนที่ 1: ความเข้าใจพื้นฐานและชนิดข้อมูล (20 คะแนน)**

**ข้อ 1. (5 คะแนน)** **คำตอบ:** ประโยชน์หลักของการเขียน "รหัสเทียม (Pseudocode)" และ "ผังงาน (Flowchart)" คือ:

1. **ช่วยในการวางแผนตรรกะและขั้นตอน (Focus on Logic):** ทำให้ผู้พัฒนาสามารถออกแบบและทบทวนลำดับการทำงานของโปรแกรมได้อย่างเป็นระบบก่อนลงมือเขียนโค้ดจริง โดยไม่ต้องกังวลเรื่องไวยากรณ์ (Syntax) ที่ซับซ้อนของภาษาโปรแกรม
2. **ช่วยในการสื่อสารและทำให้เข้าใจตรงกัน (Easy Communication):** เป็นเครื่องมือกลางที่ทำให้ทุกคนในทีม (ทั้งโปรแกรมเมอร์และผู้ที่ไม่ใช่โปรแกรมเมอร์) เห็นภาพรวมและเข้าใจการทำงานของโปรแกรมได้ตรงกัน ลดความผิดพลาดในการสื่อสาร
3. **ค้นหาข้อผิดพลาดทางตรรกะได้ง่ายขึ้น (Easier Logical Debugging):** การวางแผนที่ดีช่วยลดโอกาสเกิดข้อผิดพลาดทางตรรกะ และหากมีปัญหาก็สามารถย้อนกลับมาตรวจสอบที่แผนภาพหรือรหัสเทียมได้ง่ายกว่าการไล่โค้ดจริงที่ซับซ้อน

**ข้อ 2. (5 คะแนน)** **คำตอบ:**

* user age: **ไม่ถูกต้อง** เพราะมีเว้นวรรค (Space) อยู่ในชื่อตัวแปร
* total\_score: **ถูกต้อง** เป็นไปตามหลักการตั้งชื่อ (snake\_case)
* 2nd\_player: **ไม่ถูกต้อง** เพราะขึ้นต้นด้วยตัวเลข
* MAX\_VALUE: **ถูกต้อง** เป็นรูปแบบที่นิยมใช้สำหรับค่าคงที่ (Constant)
* if: **ไม่ถูกต้อง** เพราะเป็นคำสงวน (Keyword) ของภาษา Python

**ข้อ 3. (10 คะแนน)** **คำตอบ:** ก) ผลลัพธ์ที่โปรแกรมจะแสดงออกมาคือ: **911** ข) **เหตุผล:** เพราะฟังก์ชัน input() จะรับค่าทุกอย่างเข้ามาเป็นชนิดข้อมูล **ข้อความ (string)** เสมอ และเมื่อใช้เครื่องหมาย + กับข้อมูลชนิด string มันจะหมายถึง **การนำข้อความมาต่อกัน (Concatenation)** ไม่ใช่การบวกทางคณิตศาสตร์ โปรแกรมจึงนำข้อความ '9' มาต่อกับ '11' กลายเป็น '911'

**ตอนที่ 2: การอ่านโค้ดและคิดวิเคราะห์ (45 คะแนน)**

**ข้อ 4. (10 คะแนน)** **คำตอบ:** ลำดับการคำนวณเป็นดังนี้:

1. **วงเล็บ ()**: (15 + 5) -> 20
   * นิพจน์จะกลายเป็น: 20 \* 2 - 10 / 5 \*\* 2
2. **ยกกำลัง \*\***: 5 \*\* 2 -> 25
   * นิพจน์จะกลายเป็น: 20 \* 2 - 10 / 25
3. **คูณ \* และ หาร / (ทำจากซ้ายไปขวา)**:
   * 20 \* 2 -> 40
   * 10 / 25 -> 0.4
   * นิพจน์จะกลายเป็น: 40 - 0.4
4. **ลบ -**: 40 - 0.4 -> 39.6 **ผลลัพธ์สุดท้ายคือ 39.6**

**ข้อ 5. (10 คะแนน)** **คำตอบ:**

* **สาเหตุของความผิดพลาด:** เกิดจาก **การเรียงลำดับเงื่อนไขที่ผิด** โปรแกรมจะตรวจสอบเงื่อนไขจากบนลงล่าง เมื่อป้อนคะแนน 85 โปรแกรมจะเจอเงื่อนไขแรกคือ if score >= 50: ซึ่งเป็นจริง (85 >= 50) ทำให้โปรแกรมทำงานในบล็อกนั้นทันที (กำหนด grade = "D") และ **กระโดดออกจากโครงสร้าง if-elif-else ทั้งหมด** โดยไม่ไปตรวจสอบเงื่อนไขอื่นๆ ที่แม่นยำกว่าซึ่งอยู่ถัดไป
* **วิธีแก้ไข:** ต้องเรียงลำดับเงื่อนไข **จากค่ามากไปค่าน้อย** หรือจากเงื่อนไขที่เจาะจงที่สุดไปหาเงื่อนไขที่กว้างที่สุด ดังนี้:

รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, ภาพหน้าจอ, ออกแบบ

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง

**ข้อ 6. (10 คะแนน)** **คำตอบ:**

* **for loop**: เหมาะสมที่สุดสำหรับสถานการณ์ที่ **ทราบจำนวนรอบการทำงานที่แน่นอน** หรือต้องการวนซ้ำกับสมาชิกทุกตัวในชุดข้อมูล (Sequence)
  + **ตัวอย่าง:** ต้องการพิมพ์แม่สูตรคูณแม่ 2 (ทำ 12 รอบแน่นอน) หรือต้องการพิมพ์รายชื่อนักเรียนทุกคนในห้องที่มีข้อมูลอยู่แล้วในลิสต์
* **while loop**: เหมาะสมที่สุดสำหรับสถานการณ์ที่ **ไม่ทราบจำนวนรอบที่แน่นอน** แต่การวนซ้ำจะขึ้นอยู่กับ **เงื่อนไข (condition)** บางอย่างว่าจะทำงานต่อหรือจะหยุด
  + **ตัวอย่าง:** โปรแกรมเกมทายตัวเลข ที่จะวนไปเรื่อยๆ "ตราบใดที่" ผู้ใช้ยังทายไม่ถูก หรือโปรแกรมที่รอรับคำสั่งจากผู้ใช้ไปเรื่อยๆ จนกว่าผู้ใช้จะพิมพ์คำว่า 'exit' เพื่อจบการทำงาน

**ข้อ 7. (15 คะแนน)** **คำตอบ:**

* **การวิเคราะห์:** โปรแกรมจะประมวลผลนิพจน์ในวงเล็บก่อน แล้วจึงนำผลลัพธ์ไปประมวลผลกับตัวดำเนินการ or
  1. ประมวลผลในวงเล็บ (age >= 18 and has\_ticket):
     + age >= 18 คือ 20 >= 18 ซึ่งเป็น **True**
     + has\_ticket คือ **False**
     + ดังนั้น True and False ได้ผลลัพธ์เป็น **False**
  2. นำผลลัพธ์จากวงเล็บมาประมวลผลต่อ: False or is\_vip
     + is\_vip คือ **True**
     + ดังนั้น False or True ได้ผลลัพธ์สุดท้ายเป็น **True**
* **เหตุผล:** ที่ผลลัพธ์ยังคงเป็น True เพราะตัวดำเนินการ or จะให้ค่าเป็น True ขอเพียงแค่มีเงื่อนไขฝั่งใดฝั่งหนึ่งเป็น True ก็เพียงพอ ในกรณีนี้ แม้เงื่อนไขแรก (การมีตั๋วและอายุถึง) จะเป็นเท็จ แต่เงื่อนไขที่สอง (การเป็น VIP) เป็นจริง ทำให้เงื่อนไขโดยรวมถือว่าเป็นจริง

**ตอนที่ 3: การวิเคราะห์โปรแกรมแบบวนรอบและการออกแบบอัลกอริทึม (35 คะแนน)**

**ข้อ 8. (15 คะแนน)** **คำตอบ:** ก) **วัตถุประสงค์:** โปรแกรมนี้มีวัตถุประสงค์เพื่อคำนวณหา **ผลรวมของตัวเลขตั้งแต่ 1 ถึง 10 ที่หารด้วย 3 ไม่ลงตัว** ข) **ค่าสุดท้ายของ total\_sum:** ตัวเลขที่จะถูกนำมาบวกคือ 1, 2, 4, 5, 7, 8, 10 (ส่วนเลข 3, 6, 9 จะถูกข้ามไป) ดังนั้นผลรวมคือ 1 + 2 + 4 + 5 + 7 + 8 + 10 = 37 ค) **บทบาทของ continue:** ทำหน้าที่ **"ข้ามการทำงานที่เหลือในรอบปัจจุบัน"** แล้วไปเริ่มต้นการวนซ้ำในรอบถัดไปทันที ในโปรแกรมนี้ เมื่อ i เป็นเลขที่หารด้วย 3 ลงตัว (3, 6, 9) คำสั่ง continue จะถูกเรียก ทำให้โปรแกรมข้ามคำสั่ง total\_sum += i และไปเริ่มวนรอบของ i ตัวถัดไปเลย

**ข้อ 9. (10 คะแนน)** **คำตอบ:** จะมีการพิมพ์เครื่องหมาย \* ออกมาทั้งหมด **6 ตัว**

* **รอบที่ i = 0**: range(0) ไม่มีการวนซ้ำ -> พิมพ์ 0 ตัว
* **รอบที่ i = 1**: range(1) วน 1 ครั้ง -> พิมพ์ 1 ตัว (\*)
* **รอบที่ i = 2**: range(2) วน 2 ครั้ง -> พิมพ์ 2 ตัว (\*\*)
* **รอบที่ i = 3**: range(3) วน 3 ครั้ง -> พิมพ์ 3 ตัว (\*\*\*)
* **รวมทั้งหมด:** 0 + 1 + 2 + 3 = 6 ตัว

**ข้อ 10. (10 คะแนน)** **คำตอบ:** **รหัสเทียม (Pseudocode):**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร

เนื้อหาที่สร้างโดย AI อาจไม่ถูกต้อง