

HW1-CH1-Program ทอนเงิน

ให้ใช้กระบวนการในการแก้ปัญหา 6 ขั้นตอน เมื่อต้องการเขียนโปรแกรม “การทอนเงิน”

HW1 >  HW1.py > ...

```
1  # รับข้อมูลจากผู้ใช้
2  price = int(input("กรุณาใส่ราคา: "))
3  paid = int(input("กรุณาใส่จำนวนเงินที่จ่าย: "))
4
5  # ตรวจสอบว่าจ่ายเงินพอหรือไม่
6  if paid < price:
7      print("Error: เงินที่จ่ายไม่เพียงพอ")
8  else:
9      # คำนวณเงินทอน
10     change = paid - price
11     denominations = [1000, 500, 100, 50, 20, 10, 5, 1]
12
13     print("เงินทอน:")
14     for denom in denominations:
15         count = change // denom
16         if count > 0:
17             unit = "ใบ" if denom >= 10 else "เหรียญ"
18             print(f"{count} {unit} {denom} บาท")
19         change %= denom
20
```

ผลลัพธ์

กรุณาใส่ราคา: 2025
กรุณาใส่จำนวนเงินที่จ่าย: 3000
เงินทอน:
1 ใบ 500 บาท
4 ใบ 100 บาท
1 ใบ 50 บาท
1 ใบ 20 บาท
1 เหรียญ 5 บาท

กรุณาใส่ราคา: 2025
กรุณาใส่จำนวนเงินที่จ่าย: 2000
Error: เงินที่จ่ายไม่เพียงพอ

กระบวนการแก้ปัญหา 6 ขั้นตอนในการเขียนโปรแกรม "การทอนเงิน"

1. Identify the Problem (กำหนดปัญหา)

- ปัญหา: เราต้องการเขียนโปรแกรมที่สามารถคำนวณเงินทอนโดยแสดงจำนวนธนบัตรและเหรียญในหน่วยต่าง ๆ เช่น แบงค์ 1000, แบงค์ 500, และเหรียญ 1 เป็นต้น
-

2. Understand the Problem (ทำความเข้าใจปัญหา)

- เมื่อมีราคาสินค้าและจำนวนเงินที่จ่ายมา เราต้อง:
 - คำนวณเงินทอน = เงินที่จ่าย - ราคาสินค้า
 - หาจำนวนธนบัตรและเหรียญในแต่ละหน่วยที่ต้องใช้สำหรับเงินทอน
 - กำหนดว่าโปรแกรมจะแจ้งข้อผิดพลาดหากเงินที่จ่ายไม่เพียงพอ
-

3. Identify Alternative Ways (ระบุทางเลือกในการแก้ปัญหา)

- วิธีที่สามารถนำมาใช้:
 - ใช้ if-else และการหารตัวเลขเพื่อคำนวณจำนวนธนบัตรและเหรียญ
 - ใช้ลูป (for หรือ while) เพื่อคำนวณทีละหน่วย
 - ใช้ฟังก์ชันหรือโครงสร้างข้อมูล เช่น dictionary เพื่อจัดการกับธนบัตรและเหรียญ
-

4. Select the Best Solution (เลือกรูปแบบที่ดีที่สุด)

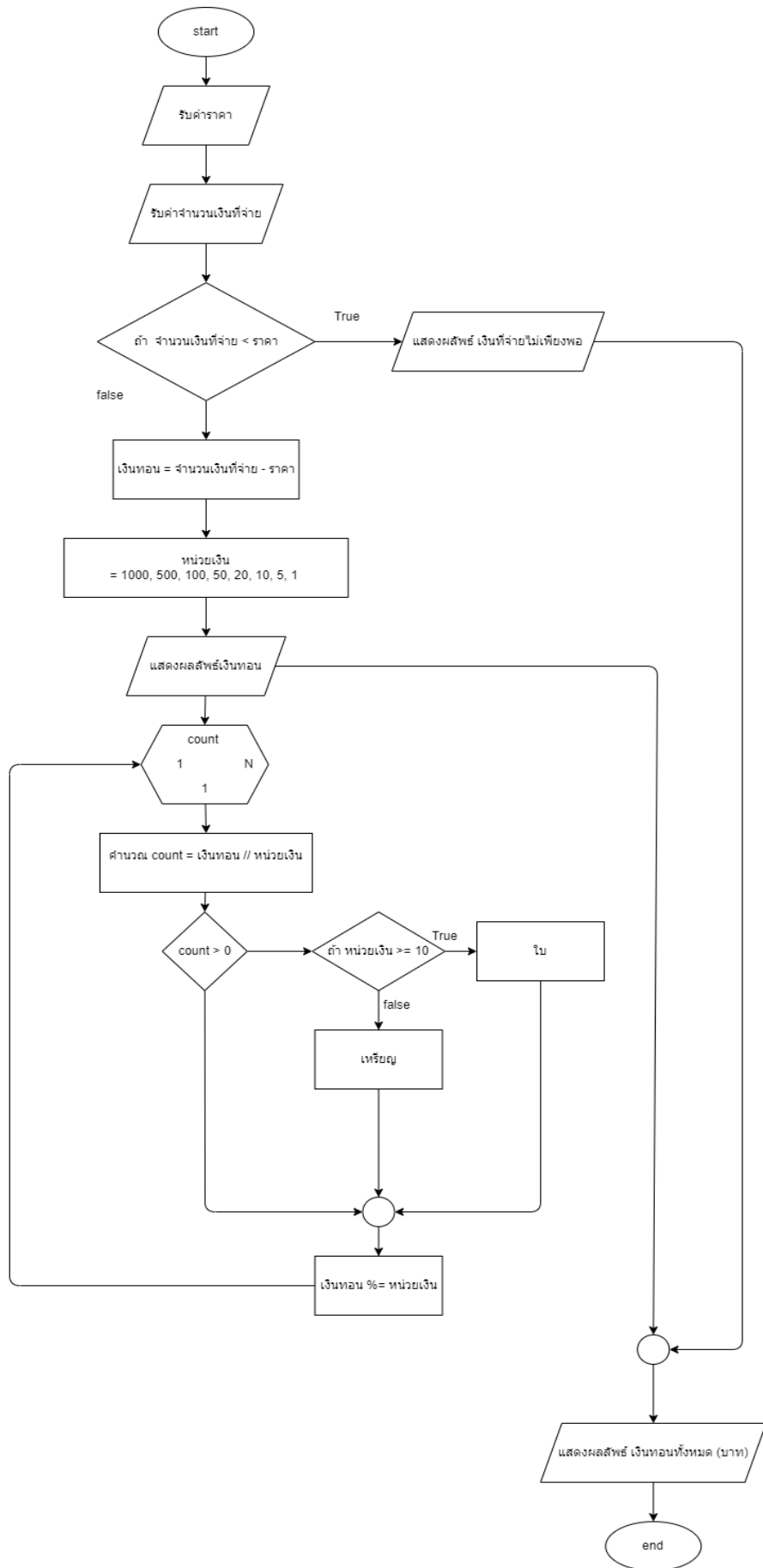
- เลือกวิธีที่ใช้ for loop กับรายการของหน่วยเงิน (denominations) เพราะอ่านง่ายและลดความซับซ้อนของโค้ด
- เพิ่มข้อความแจ้งข้อผิดพลาดกรณีเงินไม่พอเพื่อความสมบูรณ์

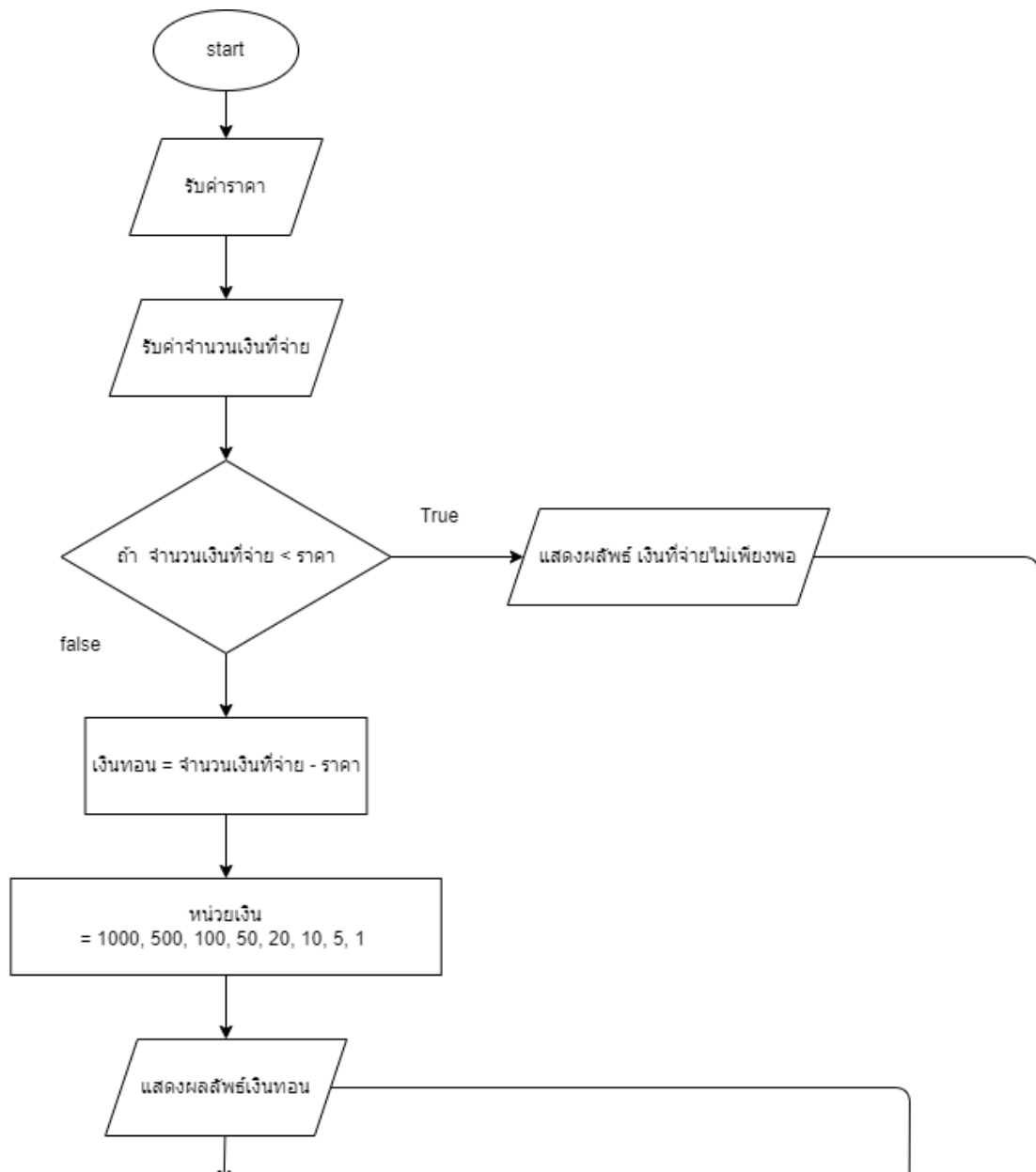
5. List Instructions (เขียนขั้นตอนการแก้ปัญหา)

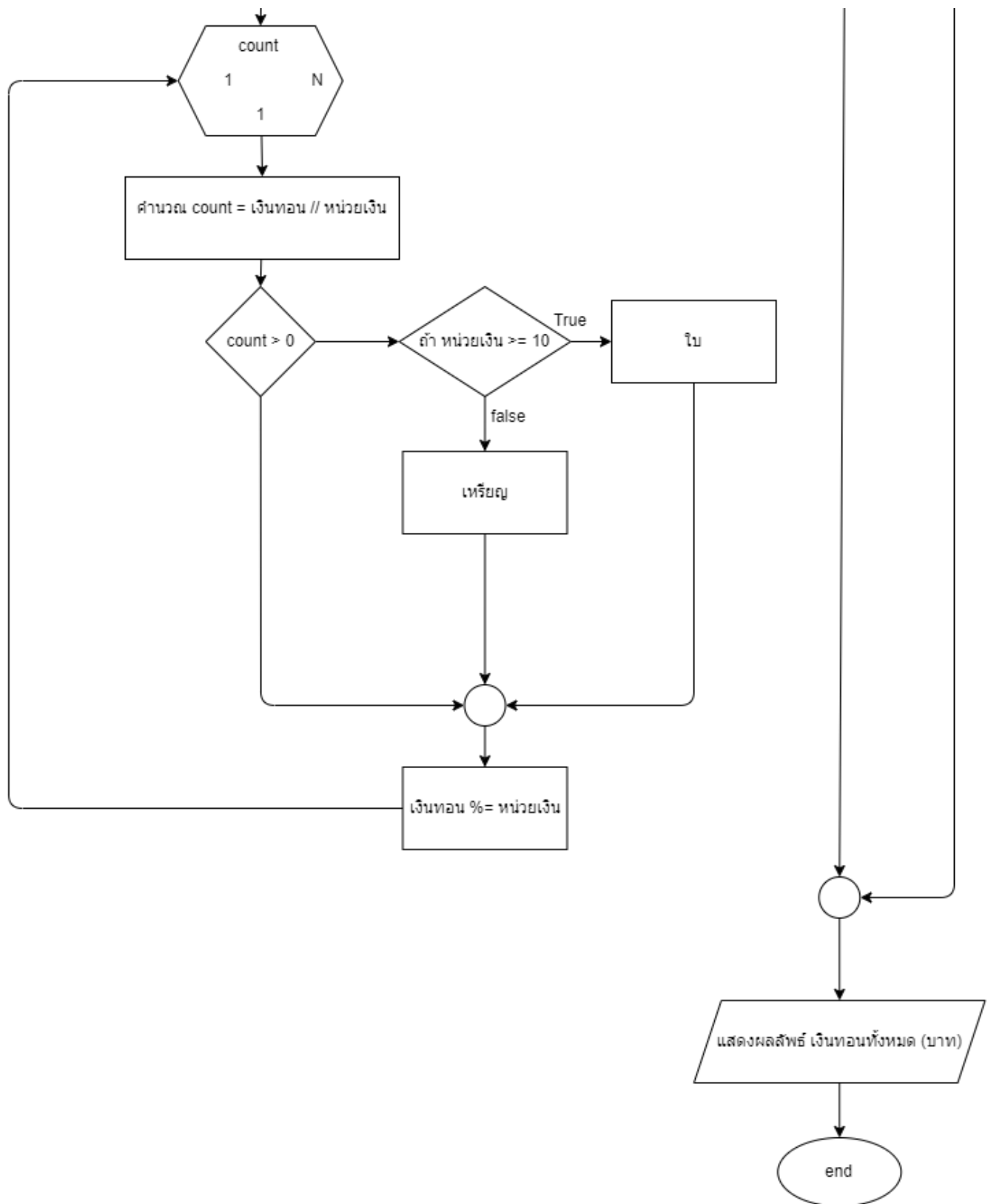
1. รับค่าราคาสินค้าและจำนวนเงินที่จ่ายจากผู้ใช้
2. ตรวจสอบว่าเงินที่จ่ายเพียงพอหรือไม่:
 - ถ้าไม่พอ: แจ้งข้อผิดพลาดและจบโปรแกรม
 - ถ้าพอ: คำนวณเงินทอน
3. ใช้รายการหน่วยเงิน (1000, 500, 100, ...) เพื่อหาจำนวนธนบัตรและเหรียญ
4. แสดงผลเงินทอนเป็นรายการของธนบัตรและเหรียญ

6. Evaluate the Solution (ตรวจสอบการทำงาน)

- ทดสอบโปรแกรมด้วยข้อมูลตัวอย่าง เช่น:
 - ราคา 350 บาท จ่าย 500 บาท ควรได้เงินทอน 150 บาท (1 ใบ 100 บาท, 1 ใบ 50 บาท)
 - ราคา 900 บาท จ่าย 1000 บาท ควรได้เงินทอน 100 บาท (1 ใบ 100 บาท)
 - กรณีจ่ายไม่พอ เช่น ราคา 400 บาท จ่าย 300 บาท ควรแจ้งข้อผิดพลาด







HW2-CH2-Analysis Chart และ Flowchart

Work1

ให้เขียน Analysis Chart และ Flowchart ของโปรแกรมคำนวณเกรดของนักศึกษาจำนวน N คน (N คือ จำนวนนักศึกษาที่รับค่าโดยผู้ใช้)

กำหนดให้

≥ 80 A, $75-79$ B+, $70-74$ B, $65-69$ C+, $60-64$ C, $55-59$ D+, $50-54$ D, < 50 F

Analysis Chart

Given Data	Required Results
<ol style="list-style-type: none">จำนวน N (จำนวนนักศึกษา)คะแนนของนักศึกษาแต่ละคน	รายการคะแนนและเกรดของนักศึกษาแต่ละคน
Required Processing	Solution Alternatives
<ol style="list-style-type: none">รับค่าคะแนนของนักศึกษาแต่ละคนตรวจสอบคะแนนและกำหนดเกรดตามเงื่อนไข:<ul style="list-style-type: none">- $\text{คะแนน} \geq 80 \rightarrow \text{A}$- $75 \leq \text{คะแนน} < 80 \rightarrow \text{B+}$- $70 \leq \text{คะแนน} < 75 \rightarrow \text{B}$- $65 \leq \text{คะแนน} < 70 \rightarrow \text{C+}$- $60 \leq \text{คะแนน} < 65 \rightarrow \text{C}$- $55 \leq \text{คะแนน} < 60 \rightarrow \text{D+}$- $50 \leq \text{คะแนน} < 55 \rightarrow \text{D}$- $\text{คะแนน} < 50 \rightarrow \text{F}$เก็บคะแนนและเกรดในลิสต์แสดงผลคะแนนและเกรดของนักศึกษาแต่ละคน	<ol style="list-style-type: none">รับค่าจำนวนนักศึกษาและคะแนนของนักศึกษาแต่ละคนใช้ if-else ในการตรวจสอบคะแนนและกำหนดเกรดใช้ Loop สำหรับดำเนินการกับนักศึกษาทั้งหมด

HW2-CH2 >  work1.py > ...

```
1  # รับจำนวน N (จำนวนนักศึกษา)
2  N = int(input("ป้อนจำนวนนักศึกษา: "))
3
4  # สร้างลิสต์สำหรับเก็บคะแนนและเกรด
5  grades = []
6
7  # วนลูปรับคะแนนและแปลงเกรด
8  for i in range(N):
9      score = float(input(f"ป้อนคะแนนของนักศึกษาคนที่ {i+1}: "))
10
11     # ตรวจสอบช่วงคะแนนและกำหนดเกรด
12     if score >= 80:
13         grade = "A"
14     elif score >= 75:
15         grade = "B+"
16     elif score >= 70:
17         grade = "B"
18     elif score >= 65:
19         grade = "C+"
20     elif score >= 60:
21         grade = "C"
22     elif score >= 55:
23         grade = "D+"
24     elif score >= 50:
25         grade = "D"
26     else:
27         grade = "F"
28
29     # เก็บเกรดในลิสต์
30     grades.append((score, grade))
31
32 # แสดงผลคะแนนและเกรด
33 print("\nผลลัพธ์คะแนนและเกรดของนักศึกษา:")
34 for i, (score, grade) in enumerate(grades):
35     print(f"นักศึกษาคนที่ {i+1}: คะแนน = {score}, เกรด = {grade}")
```


ผลลัพธ์

main.py

ป้อนจำนวนนักศึกษา : 2

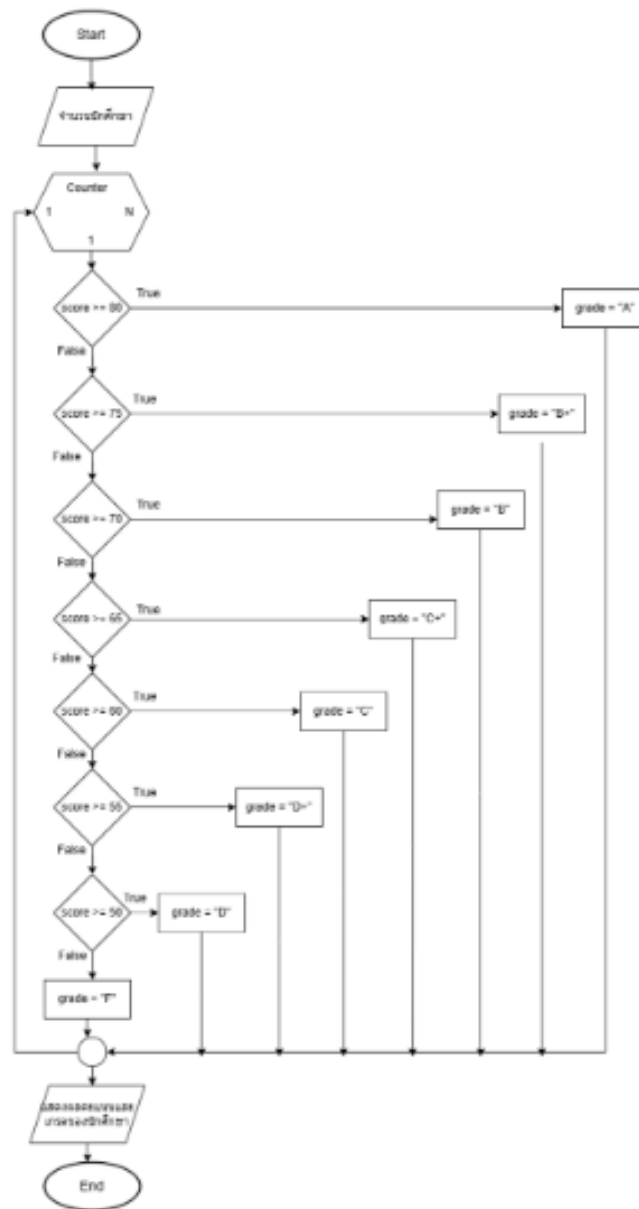
ป้อนคะแนนของนักศึกษาคนที่ 1 : 96

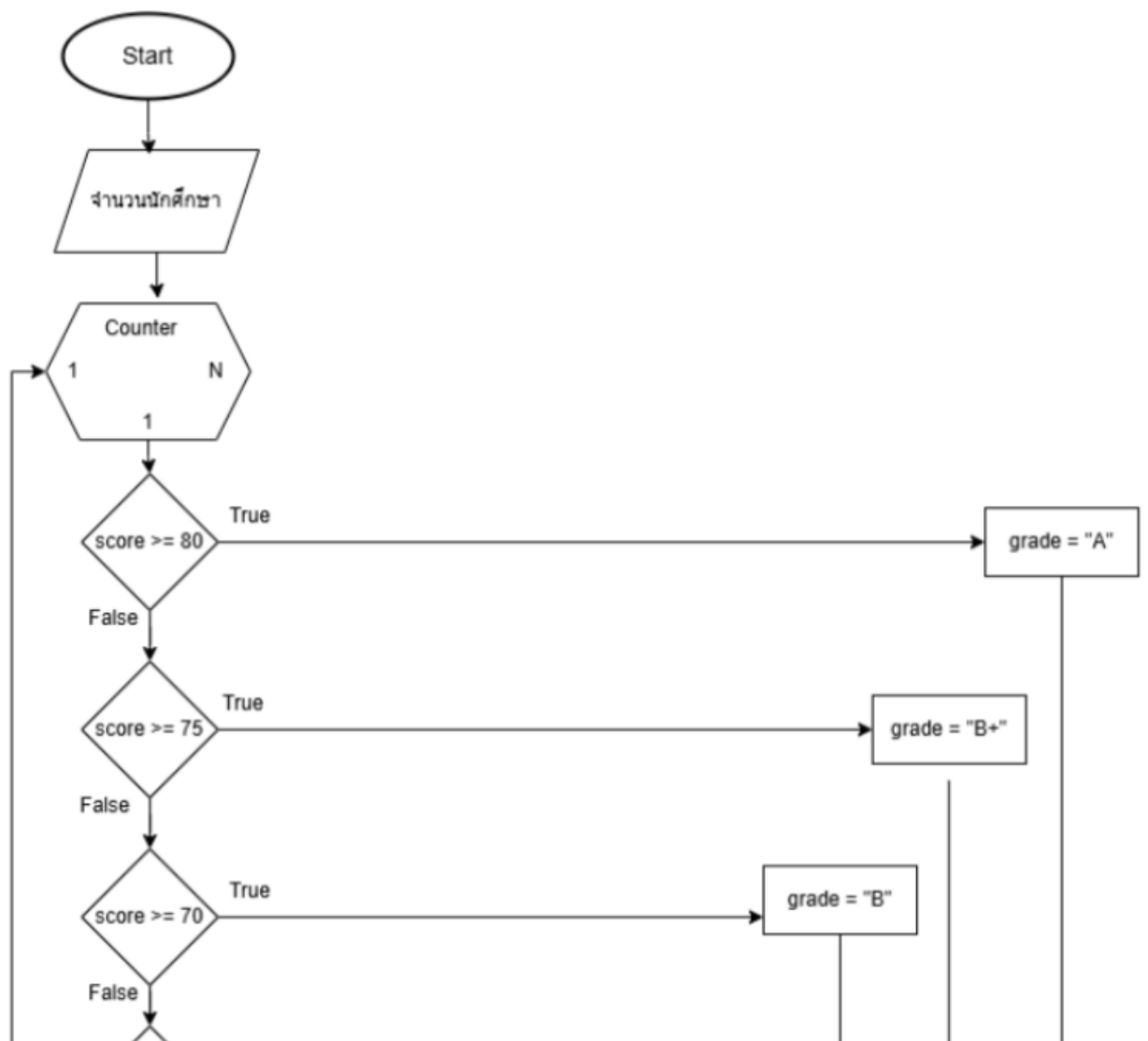
ป้อนคะแนนของนักศึกษาคนที่ 2 : 79

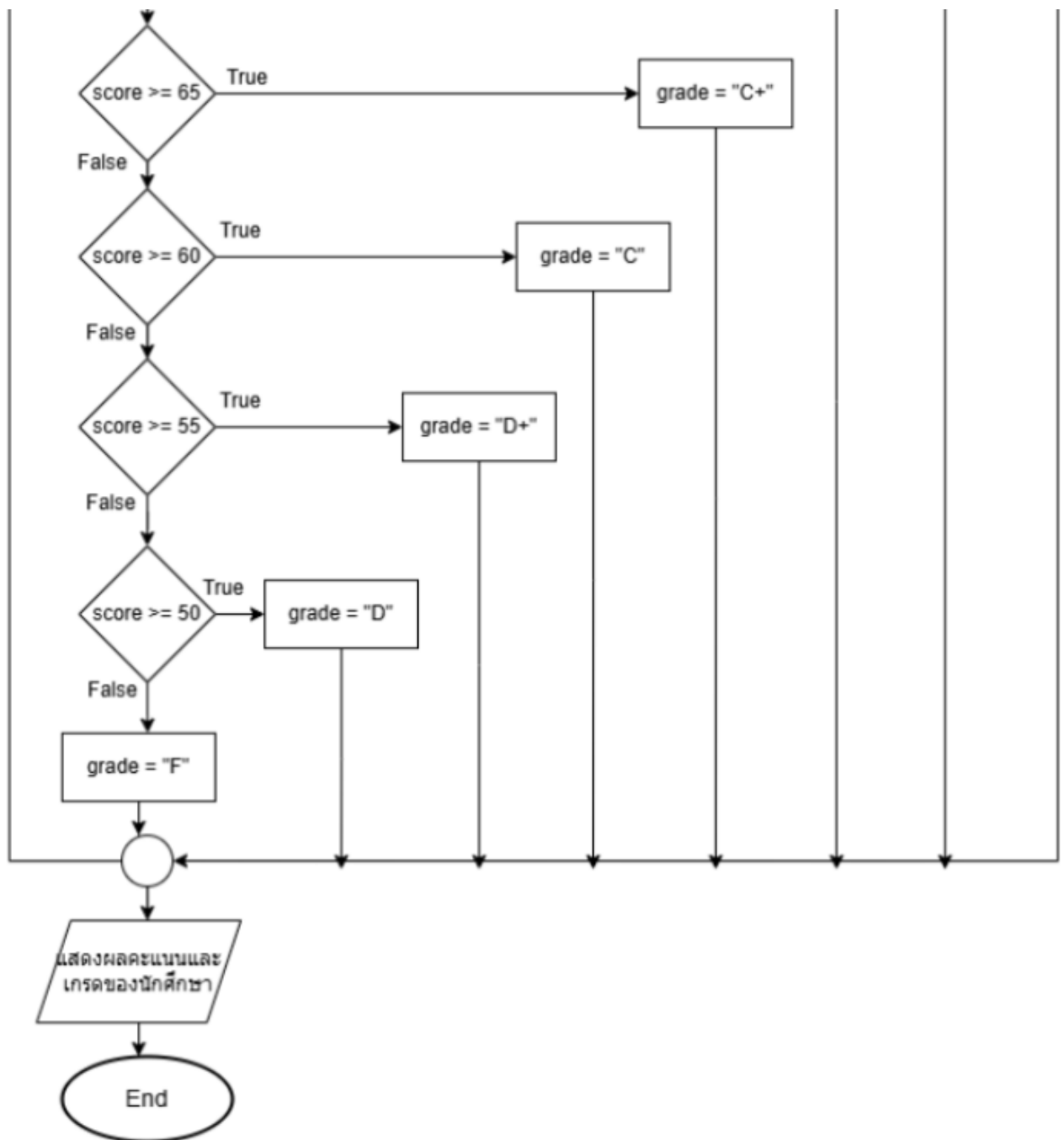
ผลลัพธ์คะแนนและเกรดของนักศึกษา :

นักศึกษาคนที่ 1 : คะแนน = 96.0 , เกรด = A

นักศึกษาคนที่ 2 : คะแนน = 79.0 , เกรด = B+







Work2

ให้เขียน Analysis Chart และ Flowchart ของโปรแกรมคำนวณเงินเดือนของพนักงานจำนวน N คน (N คือจำนวนพนักงานที่รับค่าโดยผู้ใช้)

- ถ้าชั่วโมงการทำงานในเดือนนั้นๆ ไม่เกิน 160 ชั่วโมง เงินเดือนจะถูกคำนวณโดยใช้อัตราค่าแรงตามปกติ
- ถ้าชั่วโมงการทำงานเกิน 160 ชั่วโมง 160 ชั่วโมงแรกจะใช้อัตราค่าแรงปกติ ส่วนของชั่วโมงที่เกินมาจะคิดค่าแรงโดยใช้อัตราของค่าล่วงเวลา (OT) ซึ่งเท่ากับ 1.5 เท่าของอัตราค่าแรงปกติ

Analysis Chart

Given Data	Required Results
<ol style="list-style-type: none">1. จำนวน N (จำนวนพนักงาน)2. ชั่วโมงการทำงานของพนักงาน3. อัตราค่าแรงต่อชั่วโมง	รายการเงินเดือนของพนักงานแต่ละคน
Required Processing <ol style="list-style-type: none">1. รับค่าจำนวนชั่วโมงการทำงานและอัตราค่าแรงต่อชั่วโมงของพนักงานแต่ละคน2. ตรวจสอบชั่วโมงการทำงานและคำนวณเงินเดือนตามเงื่อนไข:<ul style="list-style-type: none">- ชั่วโมงการทำงาน $\leq 160 \rightarrow$ เงินเดือน = ชั่วโมงการทำงาน \times อัตราค่าแรง- ชั่วโมงการทำงาน $> 160 \rightarrow$<ul style="list-style-type: none">- $OT = (\text{ชั่วโมงเกิน } 160) \times \text{อัตราค่าแรง} \times 1.5$- เงินเดือน = $(160 \times \text{อัตราค่าแรง}) + OT$3. เก็บจำนวนชั่วโมงและเงินเดือนในลิสต์4. แสดงผลจำนวนชั่วโมงและเงินเดือนของพนักงานแต่ละคน	Solution Alternatives <ol style="list-style-type: none">1. รับค่าชั่วโมงการทำงานและอัตราค่าแรงต่อชั่วโมงของพนักงาน2. ใช้เงื่อนไข (if-else) ในการตรวจสอบชั่วโมงการทำงานและคำนวณเงินเดือน3. ใช้ลูป (Loop) สำหรับดำเนินการกับพนักงานทั้งหมด

HW2-CH2 >  work2.py > ...

```
1  # รับจำนวน N (จำนวนพนักงาน)
2  N = int(input("ป้อนจำนวนพนักงาน: "))
3
4  # สร้างลิสต์สำหรับเก็บข้อมูลเงินเดือน
5  salaries = []
6
7  # วนลูปคำนวณเงินเดือนพนักงานแต่ละคน
8  for i in range(N):
9      hours = float(input(f"ป้อนจำนวนชั่วโมงการทำงานของพนักงานคนที่ {i+1}: "))
10     rate = float(input(f"ป้อนอัตราค่าแรงต่อชั่วโมงของพนักงานคนที่ {i+1}: "))
11
12     # คำนวณเงินเดือน
13     if hours <= 160:
14         salary = hours * rate
15     else:
16         # ชั่วโมงเกิน 160 คิด OT ที่อัตรา 1.5 เท่า
17         overtime_hours = hours - 160
18         salary = (160 * rate) + (overtime_hours * rate * 1.5)
19
20     # เก็บเงินเดือนในลิสต์
21     salaries.append((hours, salary))
22
23 # แสดงผลจำนวนชั่วโมงและเงินเดือน
24 print("\nผลลัพธ์ชั่วโมงการทำงานและเงินเดือนของพนักงาน:")
25 for i, (hours, salary) in enumerate(salaries):
26     print(f"พนักงานคนที่ {i+1}: ชั่วโมง = {hours}, เงินเดือน = {salary:.2f}")
27
```

work2.py

ป้อนจำนวนพนักงาน: 2

ป้อนจำนวนชั่วโมงการทำงานของพนักงานคนที่ 1: 160

ป้อนอัตราค่าแรงต่อชั่วโมงของพนักงานคนที่ 1: 60

ป้อนจำนวนชั่วโมงการทำงานของพนักงานคนที่ 2: 200

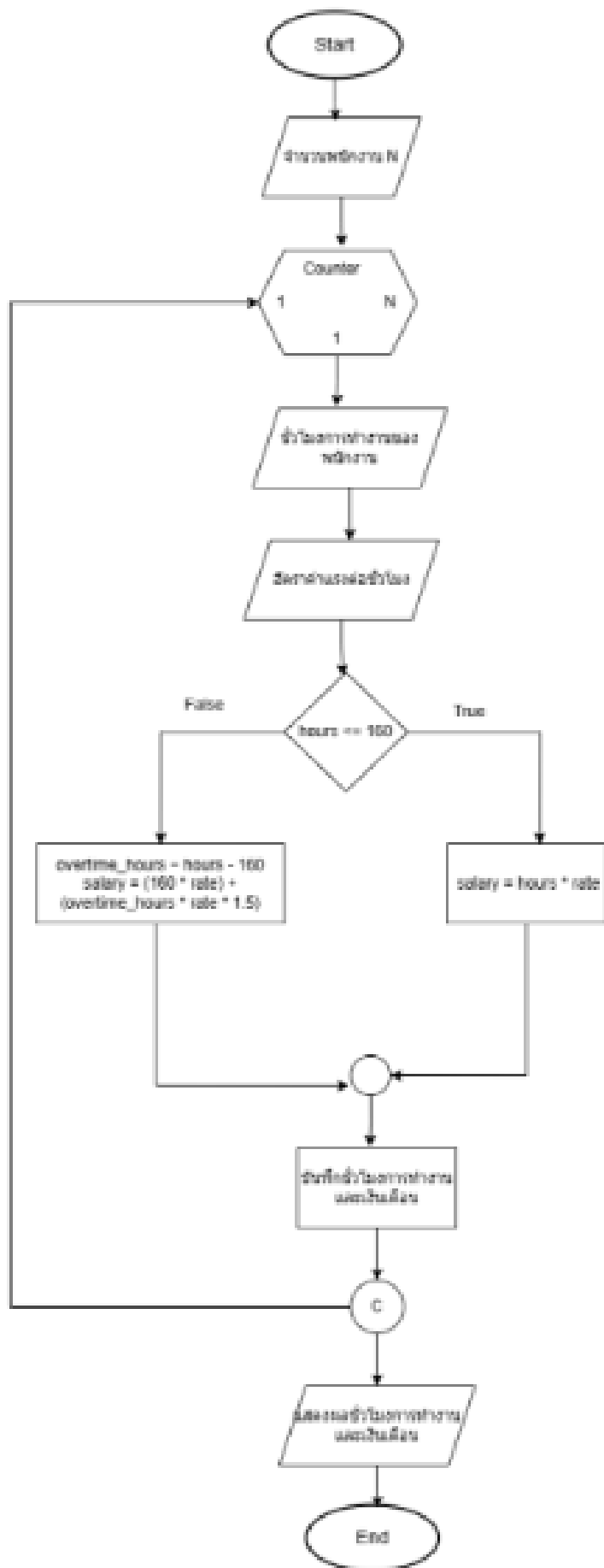
ป้อนอัตราค่าแรงต่อชั่วโมงของพนักงานคนที่ 2: 60

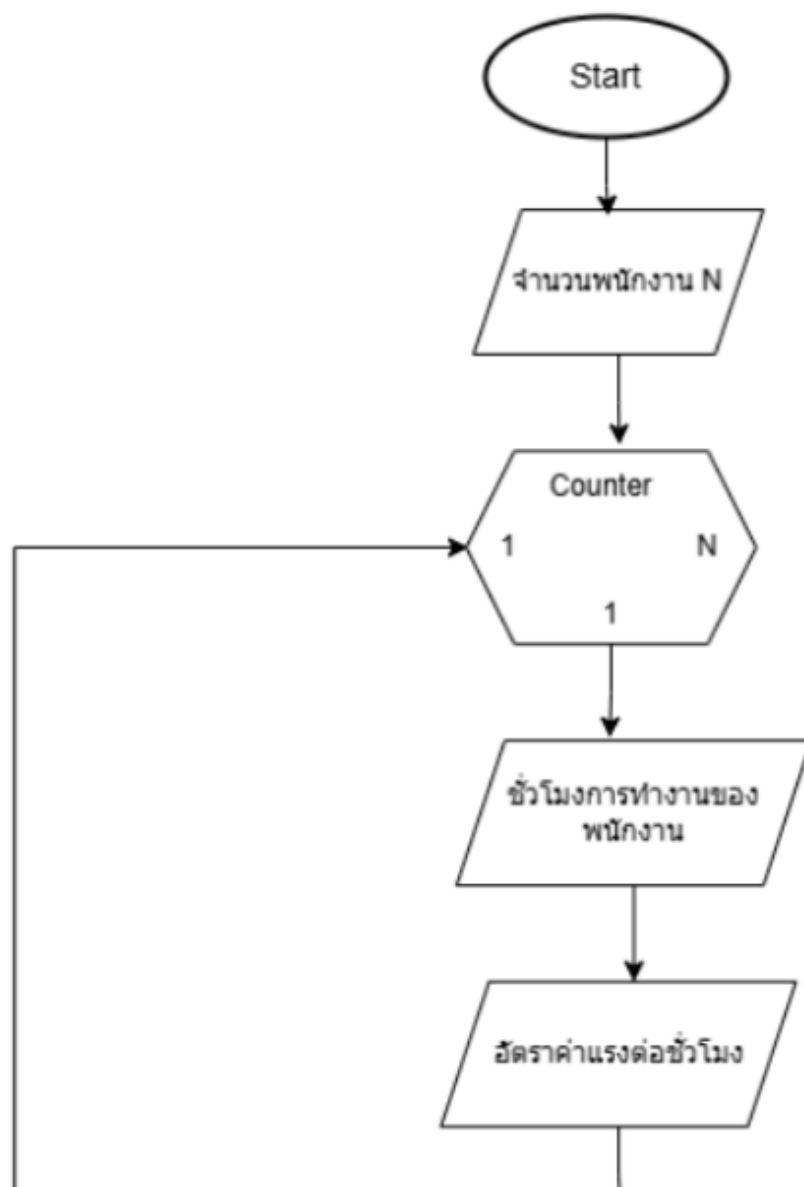
ผลลัพธ์ชั่วโมงการทำงานและเงินเดือนของพนักงาน:

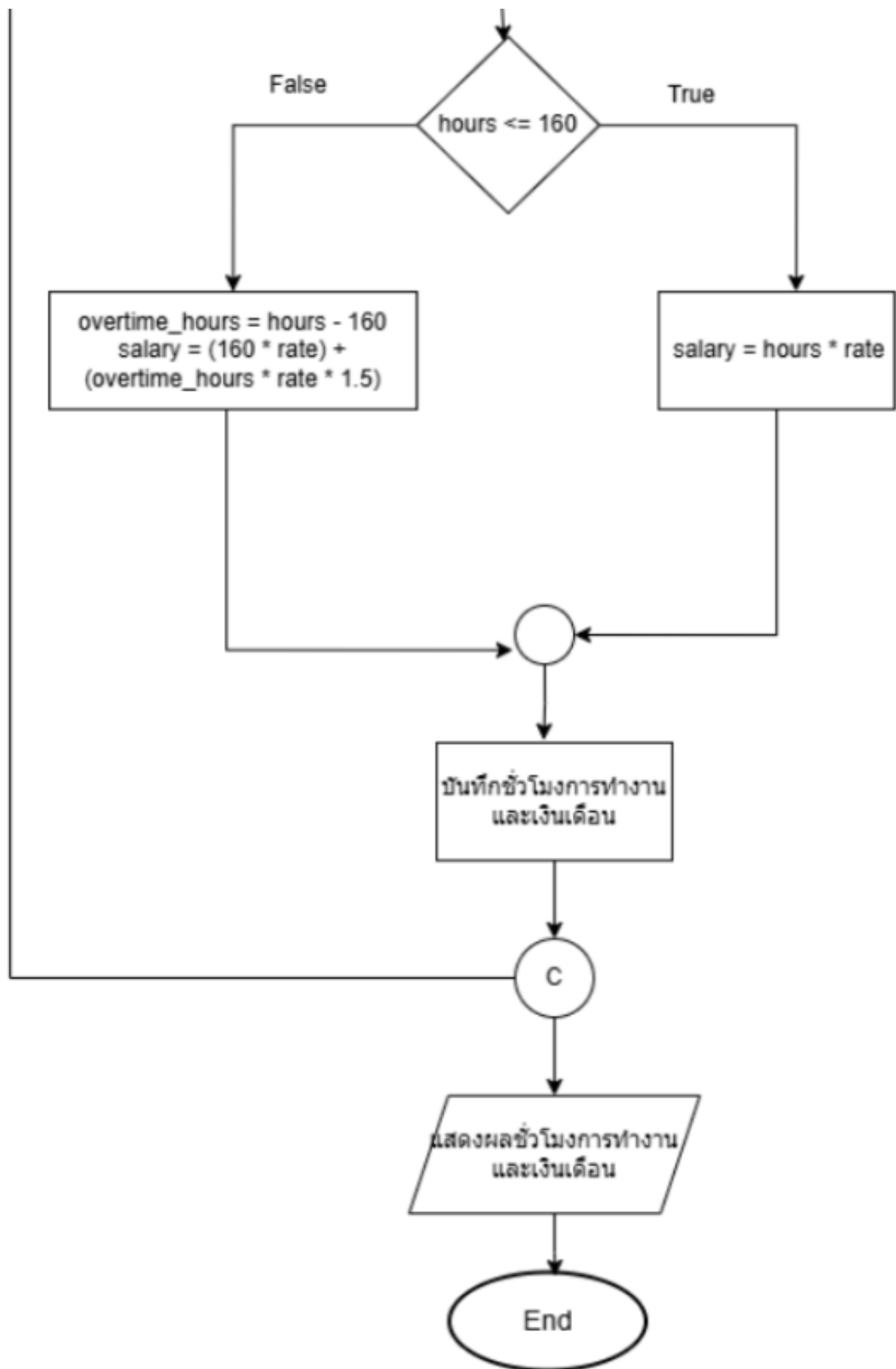
พนักงานคนที่ 1: ชั่วโมง = 160.0, เงินเดือน = 9600.00

พนักงานคนที่ 2: ชั่วโมง = 200.0, เงินเดือน = 13200.00

PS C:\Users\rawip\OneDrive\เดสก์ท็อป\ProblemSolving>







HW3-CH3-ผลรวมของเลขคู่และเลขคี่ บอกค่า Big O

เขียนโปรแกรมการหาค่าผลรวมของเลขคู่และเลขคี่ของตัวเลขที่อยู่ในช่วง 1 ถึง N เมื่อ N คือค่าที่รับจากผู้ใช้

N=10

Sum of odd number = 25

Sum of even number = 30

เมื่อ N = 500000 , Time = ?? ,Big O = ??

```
8~*P/
```

```
Enter a value for N: 20000000
```

```
Sum of even numbers = 1000000100000000
```

```
Sum of odd numbers = 10000000000000000
```

```
Time taken = 1266 ms
```

```
1266 ms
```

HW3-CH3 >  bigo.py > ...

```
1  #Big O = O(N)
2  import time
3
4  def sum_even_and_odd(N):
5      # การหาผลรวมเลขคู่และเลขคี่ในช่วง 1 ถึง N
6      sum_even = 0
7      sum_odd = 0
8      for i in range(1, N + 1):
9          if i % 2 == 0:
10             sum_even += i
11          else:
12             sum_odd += i
13      return sum_even, sum_odd
14
15  # รับค่า N จากผู้ใช้
16  N = int(input("Enter a value for N: "))
17
18  # เริ่มจับเวลา
19  start_time = time.time()
20
21  # คำนวณผลรวม
22  sum_even, sum_odd = sum_even_and_odd(N)
23
24  # สิ้นสุดการจับเวลา
25  end_time = time.time()
26
27  # แสดงผลลัพธ์
28  print(f"Sum of even numbers = {sum_even}")
29  print(f"Sum of odd numbers = {sum_odd}")
30
31  # คำนวณเวลาที่ใช้และแปลงให้เป็นจำนวนเต็ม
32  time_taken = int((end_time - start_time) * 1000)
33  print(f"Time taken = {time_taken} ms")
34  █
35
```

HW4-CH4-ตั้งโจทย์ปัญหา 1 ข้อ

ระบบคิวผู้ให้บริการธนาคาร ใช้โครงสร้างข้อมูลแบบ Queue

เงื่อนไขและข้อกำหนด

การเข้าคิว (Enqueue) - ลูกค้าสามารถเข้าคิวเพื่อรอใช้บริการ

การออกจากคิว (Dequeue) - เมื่อลูกค้าเสร็จสิ้นการให้บริการ พวกเขาควรถูกนำออกจากคิว

จัดการกับสถานการณ์คิวเต็มและคิวว่าง - ให้โปรแกรมสามารถตรวจสอบและจัดการกับสถานการณ์ที่คิวเต็มหรือคิวว่างได้

ตรวจสอบจำนวนผู้ใช้ในคิว - โปรแกรมควรมีฟังก์ชันเพื่อตรวจสอบจำนวนลูกค้าที่กำลังรอในคิว

แสดงผลลัพธ์-

โปรแกรมควรสามารถแสดงหมายเลขของผู้ใช้บริการปัจจุบันและหมายเลขของผู้ใช้บริการถัดไปทุกครั้งที่มีการเรียกดูหรืออัปเดตคิว

ให้เขียนโปรแกรมพัฒนาระบบจัดการข้อมูลนักศึกษาโดยมีฟังก์ชันการทำงานดังนี้

1. ใช้โครงสร้างข้อมูลแบบ Linked List ในการเก็บข้อมูลนักศึกษา
2. ข้อมูลที่ต้องจัดเก็บสำหรับนักศึกษาแต่ละคน
 - รหัสนักศึกษา (Student ID) เป็นค่าไม่ซ้ำ
 - ชื่อ-นามสกุล (Name)
 - คะแนนเฉลี่ยสะสม (GPA) ต้องอยู่ในช่วง 0.00 - 4.00
3. ระบบต้องมีฟังก์ชันการทำงานดังนี้
 - เพิ่มนักศึกษา (Add Student) เพิ่มข้อมูลนักศึกษาใหม่
 - ลบนักศึกษา (Remove Student) ลบข้อมูลนักศึกษาตามรหัสนักศึกษา

- ค้นหานักศึกษา (Search Student) แสดงข้อมูลนักศึกษาที่มีรหัสตรงกับที่ระบุ
- แสดงรายชื่อนักศึกษาทั้งหมด (Display All Students) แสดงข้อมูลนักศึกษาทั้งหมดในระบบ
- คำนวณคะแนนเฉลี่ยรวม (Calculate Average GPA) คำนวณและแสดงคะแนนเฉลี่ยรวมของนักศึกษาทั้งหมดในระบบ

4. เงื่อนไขเพิ่มเติม

- ห้ามเพิ่มข้อมูลนักศึกษาที่มีรหัสซ้ำกับในระบบ
- หากไม่มีข้อมูลนักศึกษาในระบบ ให้แสดงข้อความ "ไม่มีข้อมูลนักศึกษาในระบบ"
- หากพยายามค้นหาหรือลบข้อมูลนักศึกษาที่ไม่มีในระบบ ให้แสดงข้อความ "ไม่พบนักศึกษาที่ระบุ"

SubTest #1 : Problem Analysis Chart, Flowchart, Test case, และ Program

จงเขียน Problem Analysis Chart, Flowchart, Test case, และ Program ของโจทย์ด้านล่างนี้

โจทย์:

เมื่อนักศึกษาถูกมอบหมายให้ออกแบบและเขียนโปรแกรมสำหรับการวางแผนเงินออมที่สามารถเลือกได้ว่าจะออมนรายวันหรือรายเดือน ตามเป้าหมายจำนวนเงินและจำนวนปีที่ผู้ใช้งานระบุ

Given Data 1. จำนวนเงินทั้งหมดที่อยากเก็บ 2. ระยะเวลาที่อยากเก็บเงิน (กี่ปี) 3. จะออมนรายวันหรือรายเดือน	Required Results - จำนวนเงินที่ต้องออมนต่อวัน/เดือน - แสดงผลการคำนวณตามเป้าหมายที่ผู้ใช้กำหนด
Required Processing 1. รับข้อมูลจากผู้ใช้ เช่น เป้าหมาย, ระยะเวลา, และ ความถี่ 2. ตรวจสอบว่าเลือกออมแบบรายวันหรือรายเดือน - ถ้าออมนรายวัน = $(\text{เป้าหมาย}) \div (\text{ระยะเวลา} \times 365)$ - ถ้าออมนรายเดือน = $(\text{เป้าหมาย}) \div (\text{ระยะเวลา} \times 12)$ 3. คำนวณจำนวนเงินที่ต้องออม 4. แสดงผลลัพธ์ให้ผู้ใช้ดู	Solution Alternatives 1. ใช้คำสั่ง if-else ในการตรวจสอบความถี่ 2. ใช้สูตรคำนวณเงินออมในแต่ละกรณี (รายวัน/รายเดือน)

SubTest1 > test1.py > ...

```
1  # โปรแกรมคำนวณเงินออม
2  print("โปรแกรมคำนวณเงินออม")
3
4  # รับข้อมูลจากผู้ใช้
5  goal = float(input("กรุณกรอกเป้าหมายเงินออม (บาท): "))
6  years = int(input("กรุณกรอระยะเวลา (ปี): "))
7  frequency = input("เลือกความถี่การออม (วัน/เดือน): ").lower()
8
9  # คำนวณเงินออม
10 if frequency == "วัน":
11     savings = goal / (years * 365)
12     print(f"คุณต้องออมเงิน {savings:.2f} บาทต่อวัน")
13 elif frequency == "เดือน":
14     savings = goal / (years * 12)
15     print(f"คุณต้องออมเงิน {savings:.2f} บาทต่อเดือน")
16 else:
17     print("ความถี่ที่กรอกไม่ถูกต้อง กรุณาเลือก วัน หรือ เดือน")
18
```

copy

โปรแกรมคำนวณเงินออม

กรุณกรอกเป้าหมายเงินออม (บาท): 500000

กรุณกรอระยะเวลา (ปี): 2

เลือกความถี่การออม (วัน/เดือน): วัน

คุณต้องออมเงิน 684.93 บาทต่อวัน

Press any key to continue . . .

copy

โปรแกรมคำนวณเงินออม

กรุณกรอกเป้าหมายเงินออม (บาท): 50000

กรุณกรอระยะเวลา (ปี): 2

เลือกความถี่การออม (วัน/เดือน): เดือน

คุณต้องออมเงิน 2083.33 บาทต่อ เดือน

Press any key to continue . . .

Test Case

เป้าหมาย: 120,000 บาท

ระยะเวลา: 1 ปี

ความถี่: รายเดือน

ต้องออม 10,000 บาทต่อเดือน

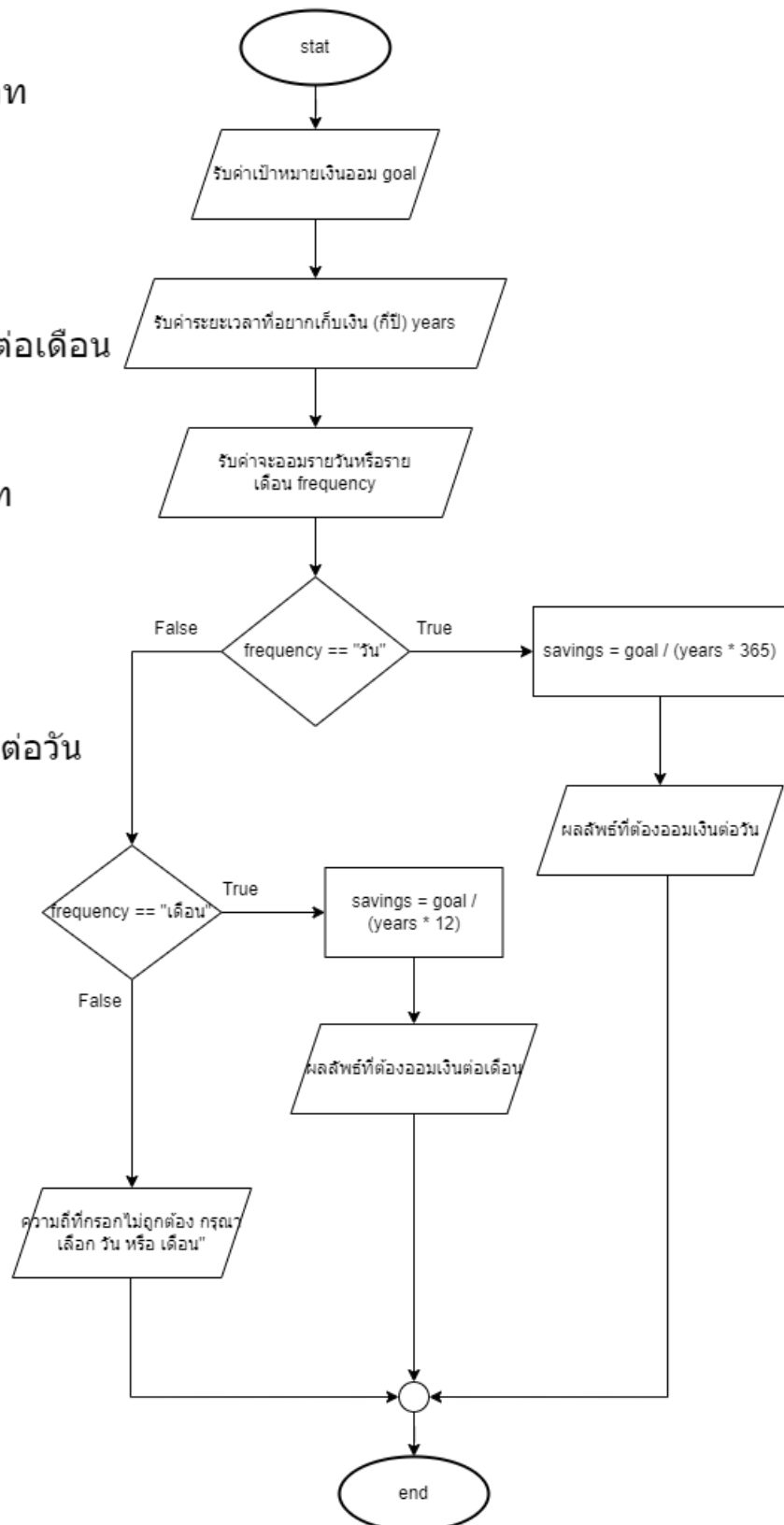
เป้าหมาย: 120,000 บาท

ระยะเวลา: 1 ปี

ความถี่: วัน

ต้องออม 328.77 บาทต่อวัน

Flowchart



SubTest#2: โปรแกรมบันทึกการเผาผลาญแคลอรี

TestSubTest > Test_6406022620070_1.py > ...

```
1 total_calories = 0
2
3 while True:
4     activity = input("กิจกรรม \n1.วิ่ง \n2.ปั่นจักรยาน \n3.ว่ายน้ำ \n4.จบการทำงาน \nป้อนกิจกรรม: ")
5
6     if activity == '4':
7         break
8
9     try:
10        duration = int(input("ป้อนระยะเวลาในการทำกิจกรรม (นาที): "))
11    except ValueError:
12        print("กรุณาป้อนตัวเลขสำหรับระยะเวลา!")
13        continue
14
15    if activity == '1' or 'วิ่ง':
16        calories = duration * 10
17    elif activity == '2' or 'ปั่นจักรยาน':
18        calories = duration * 8
19    elif activity == '3' or 'ว่ายน้ำ':
20        calories = duration * 5
21    else:
22        print("\nกิจกรรมที่ป้อนไม่ถูกต้อง กรุณาลองอีกครั้ง!\n")
23        continue
24
25    total_calories += calories
26    print(f"\nคุณเผาผลาญแคลอรีไป {calories} แคลอรีจากกิจกรรม: {activity}")
27    print(f"แคลอรีที่สะสมทั้งหมด: {total_calories} แคลอรี\n")
28    print("-----")
29
30 print("\n===== สรุปผลรวม =====")
31 print(f"แคลอรีที่เผาผลาญรวมทั้งหมด: {total_calories} แคลอรี")
32 print("=====")
33
```


กิจกรรม

1. รุ่ง
2. บั่นจักรยาน
3. ว่ายน้ำ
4. จมการทำงาน

ป้อนกิจกรรม: 1

ป้อนระยะเวลาในการทำกิจกรรม (นาที): 30

คุณ ผาผลาญแคลอรีไป 300 แคลอรีจากกิจกรรม: 1

แคลอรีที่สะสมทั้งหมด: 300 แคลอรี

กิจกรรม

1. รุ่ง
2. บั่นจักรยาน
3. ว่ายน้ำ
4. จมการทำงาน

ป้อนกิจกรรม: 2

ป้อนระยะเวลาในการทำกิจกรรม (นาที): 60

คุณ ผาผลาญแคลอรีไป 600 แคลอรีจากกิจกรรม: 2

แคลอรีที่สะสมทั้งหมด: 900 แคลอรี

กิจกรรม

1. รุ่ง
2. บั่นจักรยาน
3. ว่ายน้ำ
4. จมการทำงาน

ป้อนกิจกรรม: 3

ป้อนระยะเวลาในการทำกิจกรรม (นาที): 50

คุณ ผาผลาญแคลอรีไป 500 แคลอรีจากกิจกรรม: 3

แคลอรีที่สะสมทั้งหมด: 1400 แคลอรี

กิจกรรม

1. รุ่ง
2. บั่นจักรยาน
3. ว่ายน้ำ
4. จมการทำงาน

ป้อนกิจกรรม: 4

===== สรุปผลรวม =====

แคลอรีที่ ผาผลาญรวมทั้งหมด: 1400 แคลอรี

=====

Analysis Chart

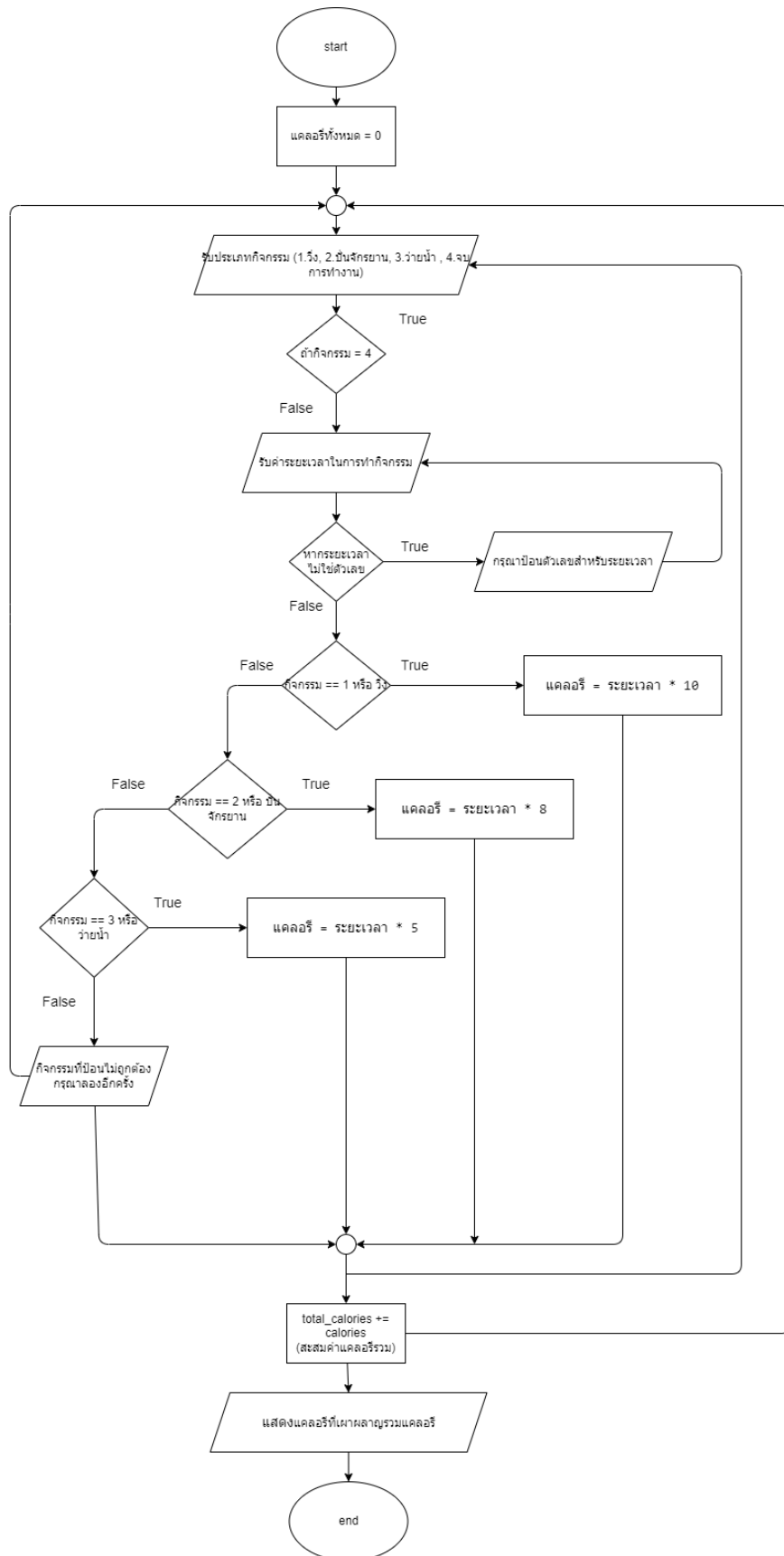
<p>Given Data</p> <ol style="list-style-type: none"> ประเภทการออกกำลังกาย <ul style="list-style-type: none"> - Running (วิ่ง) = 10 แคลอรีต่อนาที - Cycling (ปั่นจักรยาน) = 8 แคลอรีต่อนาที - Swimming (ว่ายน้ำ) = 5 แคลอรีต่อนาที ระยะเวลาในการออกกำลังกาย (นาที) 	<p>Required Results</p> <ul style="list-style-type: none"> - แคลอรีที่เผาผลาญในแต่ละรอบของการทำกิจกรรม - แคลอรีสะสมทั้งหมดจากกิจกรรมทั้งหมด - แคลอรีต่อนาที แสดงผลลัพธ์แคลอรีสะสมเมื่อจบโปรแกรม
<p>Required Processing</p> <ol style="list-style-type: none"> รับข้อมูลประเภทการออกกำลังกาย (1 = วิ่ง, 2 = ปั่นจักรยาน, 3 = ว่ายน้ำ, 4 = จบการทำงาน) ตรวจสอบความถูกต้องของข้อมูลที่ป้อนมา (กิจกรรมและระยะเวลา) รับข้อมูลระยะเวลา (นาที) คำนวณแคลอรีที่เผาผลาญโดยใช้สูตร: <ul style="list-style-type: none"> - วิ่ง: แคลอรี = ระยะเวลา * 10 - ปั่นจักรยาน: แคลอรี = ระยะเวลา * 8 - ว่ายน้ำ: แคลอรี = ระยะเวลา * 5 เพิ่มค่าแคลอรีที่คำนวณได้ลงในตัวแปรสะสม (total_calories) แสดงผลลัพธ์แคลอรีที่เผาผลาญในแต่ละรอบของกิจกรรม แสดงผลลัพธ์แคลอรีสะสมเมื่อสิ้นสุดการทำงาน 	<p>Solution Alternatives</p> <ol style="list-style-type: none"> ใช้ if-elif-else สำหรับการตรวจสอบประเภทกิจกรรม ใช้ while loop เพื่อตรวจสอบและทำงานซ้ำจนกว่าจะเลือก "จบการทำงาน" ใช้ตัวแปร total_calories เพื่อสะสมแคลอรี

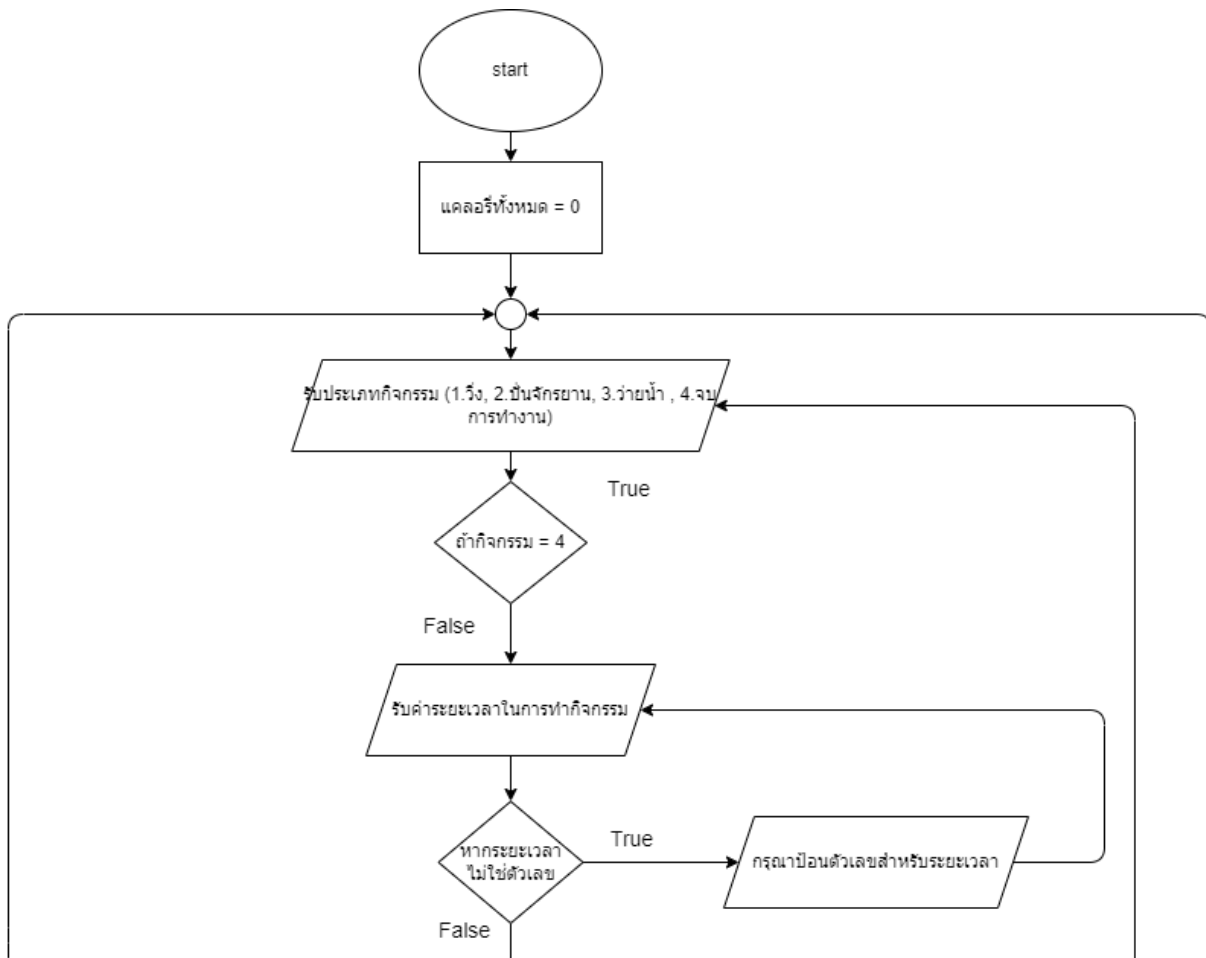
IPO Chart

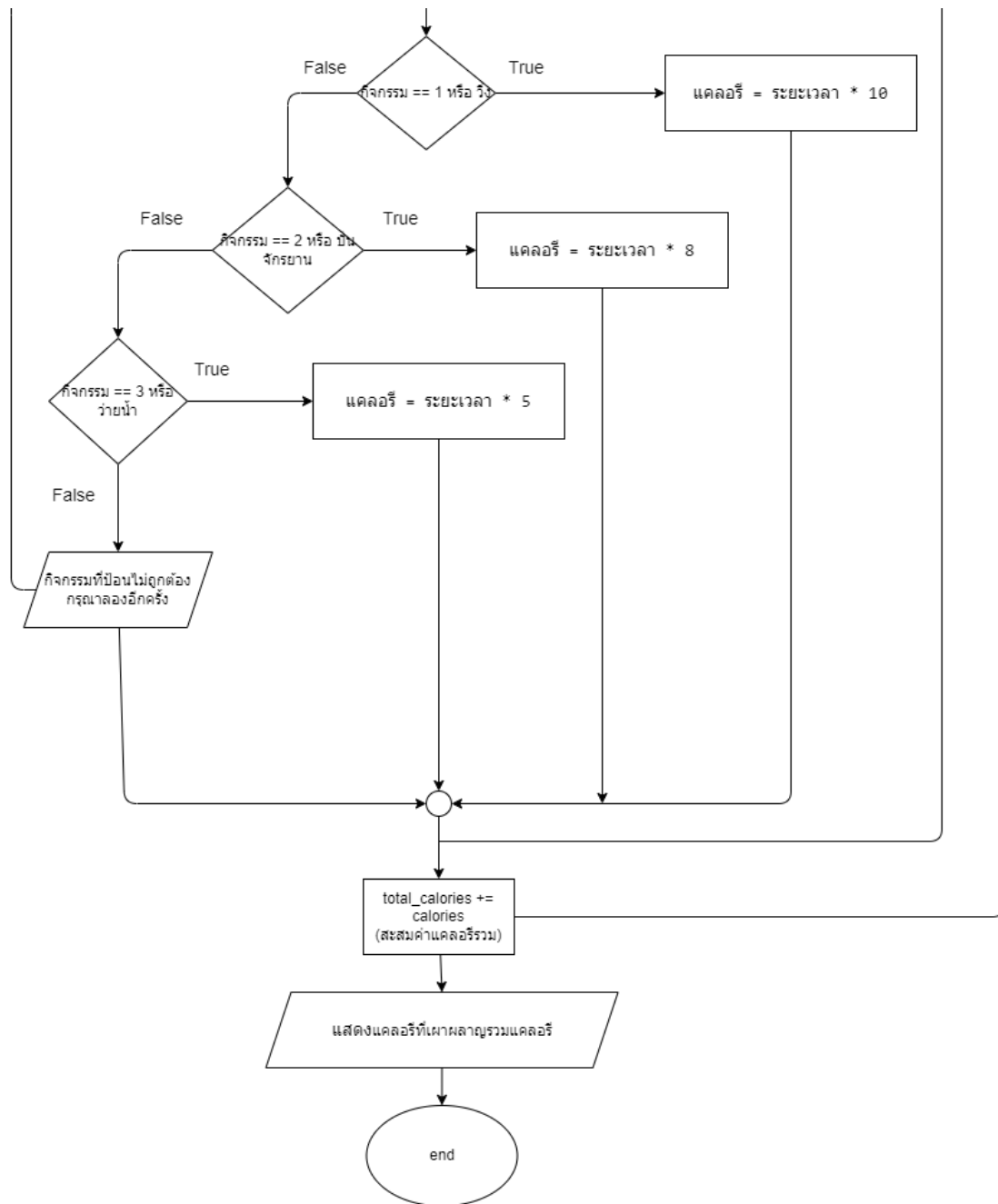
Input	Processing	Module Reference	Output
1. ประเภทกิจกรรม (1 = รีด, 2 = ปั่นจักรยาน, 3 = วายน้ำ, 4 = จบการทำงาน)	1. รับประเภทกิจกรรมจากผู้ใช้	InputActivity	แคลอรีที่เผาผลาญจากแต่ละกิจกรรม
2. ระยะเวลาในการทำกิจกรรม (นาที)	2. ตรวจสอบประเภทกิจกรรมและระยะเวลา	ValidateInput	แคลอรีสะสมรวมทั้งหมด
	3. คำนวณแคลอรีโดยใช้สูตร (10, 8, 5 แคลอรีต่อนาทีตามประเภทกิจกรรม)	CalculateCalories	
	4. สะสมค่าแคลอรีทั้งหมด	UpdateTotalCalories	
	5. แสดงผลลัพธ์แคลอรีในแต่ละรอบ และแสดงผลลัพธ์แคลอรีสะสมทั้งหมดเมื่อจบโปรแกรม	DisplayCalories, DisplayTotal	

คำอธิบาย:

- **Input:** ข้อมูลที่ผู้ใช้ป้อน ได้แก่ ประเภทกิจกรรมและระยะเวลาในการทำกิจกรรม
- **Processing:** ขั้นตอนที่โปรแกรมดำเนินการ เช่น การตรวจสอบข้อมูล การคำนวณ และการสะสมค่าแคลอรี
- **Module Reference:** ฟังก์ชันหรือโมดูลที่เกี่ยวข้องในกระบวนการทำงาน
- **Output:** ผลลัพธ์ที่แสดงให้ผู้ใช้ เช่น แคลอรีที่เผาผลาญในแต่ละรอบและแคลอรีสะสมทั้งหมด







Test Case 1: กรณีปกติ

Test Case	Input	Expected Output
TC1-1	กิจกรรม = 1, ระยะเวลา = 30	แคลอรีที่เผาผลาญ = 300 แคลอรี (วิ่ง), แคลอรีสะสม = 300 แคลอรี
TC1-2	กิจกรรม = 2, ระยะเวลา = 15	แคลอรีที่เผาผลาญ = 120 แคลอรี (ปั่นจักรยาน), แคลอรีสะสม = 420 แคลอรี
TC1-3	กิจกรรม = 3, ระยะเวลา = 20	แคลอรีที่เผาผลาญ = 100 แคลอรี (ว่ายน้ำ), แคลอรีสะสม = 520 แคลอรี
TC1-4	กิจกรรม = 4	จบการทำงาน, แสดงแคลอรีสะสมทั้งหมด = 520 แคลอรี

Test Case 2: การป้อนข้อมูลผิด

Test Case	Input	Expected Output
TC2-1	กิจกรรม = 5	"กิจกรรมที่ป้อนไม่ถูกต้อง กรุณาลองอีกครั้ง" และวนกลับไปเริ่มต้นรูป
TC2-2	กิจกรรม = 1, ระยะเวลา = abc	"กรุณาป้อนตัวเลขสำหรับระยะเวลา!" และวนกลับไปเริ่มต้นรูป

Test Case 3: การป้อนข้อมูลขอบเขตพิเศษ

Test Case	Input	Expected Output
TC3-1	กิจกรรม = 1, ระยะเวลา = 0	แคลอรีที่เผาผลาญ = 0 แคลอรี (วิ่ง), แคลอรีสะสมไม่มีการเปลี่ยนแปลง
TC3-2	กิจกรรม = 3, ระยะเวลา = -10	"กรุณาป้อนตัวเลขสำหรับระยะเวลา!" และวนกลับไปเริ่มต้นรูป

Test Case 4: กรณีทำงานต่อเนื่องหลายรอบ

Test Case	Input	Expected Output
TC4-1	รอบที่ 1: กิจกรรม = 1, ระยะเวลา = 10	แคลอรีที่เผาผลาญ = 100 แคลอรี, แคลอรีสะสม = 100 แคลอรี
	รอบที่ 2: กิจกรรม = 2, ระยะเวลา = 20	แคลอรีที่เผาผลาญ = 160 แคลอรี, แคลอรีสะสม = 260 แคลอรี
	รอบที่ 3: กิจกรรม = 3, ระยะเวลา = 15	แคลอรีที่เผาผลาญ = 75 แคลอรี, แคลอรีสะสม = 335 แคลอรี
	รอบที่ 4: กิจกรรม = 4	แสดงแคลอรีสะสมทั้งหมด = 335 แคลอรี, จบการทำงาน

Test Case 5: กรณีปิดโปรแกรมทันที

Test Case	Input	Expected Output
TC5-1	กิจกรรม = 4	แสดงแคลอรีสะสมทั้งหมด = 0 แคลอรี, จบการทำงาน

เพิ่มเติม

คำนวณ BMR : โปรแกรมคำนวณอัตราการเผาผลาญแคลอรี

สถานการณ์: คุณเป็นนักพัฒนาซอฟต์แวร์ที่ต้องการสร้างโปรแกรมเพื่อช่วยผู้ใช้คำนวณอัตราการเผาผลาญแคลอรีต่อวัน โดยพิจารณาจากน้ำหนัก, ส่วนสูง, อายุ, และเพศของพวกเขา โปรแกรมนี้จะใช้สูตร Mifflin-St Jeor ซึ่งเป็นหนึ่งในสูตรที่แม่นยำและนิยมใช้มากที่สุดในการคำนวณการเผาผลาญพลังงานต่ำสุด (Basal Metabolic Rate - BMR).

สูตร Mifflin-St Jeor:

- สำหรับผู้ชาย: $BMR = 10 * \text{weight (kg)} + 6.25 * \text{height (cm)} - 5 * \text{age (y)} + 5$
- สำหรับผู้หญิง: $BMR = 10 * \text{weight (kg)} + 6.25 * \text{height (cm)} - 5 * \text{age (y)} - 161$

ข้อมูลนำเข้า:

- น้ำหนัก (กิโลกรัม)
- ส่วนสูง (เซนติเมตร)
- อายุ (ปี)
- เพศ ('ชาย' หรือ 'หญิง')

กระบวนการ:

- รับข้อมูลน้ำหนัก, ส่วนสูง, อายุ และเพศจากผู้ใช้
- คำนวณ BMR โดยใช้สูตรที่เหมาะสมตามเพศของผู้ใช้

ผลลัพธ์:

- แสดงอัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR)

Analysis Chat

Given Data -รับค่าน้ำหนัก -รับค่าส่วนสูง -รับค่า เพศ (ชาย หรือ หญิง) - รับค่า อายุ	Required Results <ul style="list-style-type: none">• แสดงอัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR)
Required Processing 1. รับข้อมูลจากผู้ใช้ เช่น น้ำหนัก ส่วนสูง เพศ และ อายุ 2. ตรวจสอบว่าเป็น ชาย หรือหญิง - ผู้ชาย: $BMR = 10 * \text{weight (kg)} + 6.25 * \text{height (cm)} - 5 * \text{age (y)} + 5$ - ผู้หญิง: $BMR = 10 * \text{weight (kg)} + 6.25 * \text{height (cm)} - 5 * \text{age (y)} - 161$ 3. คำนวณ BMR 4.แสดงผลลัพธ์ให้ผู้ใช้ดู	Solution Alternatives 1.ใช้ if-else ในการตรวจสอบว่าชายหรือ หญิง 2.ใช้สูตรคำนวณBMR ชายหรือหญิง

TestSubTest >  BMR.py > ...

```
1  print("คำนวณอัตราการเผาผลาญแคลอรี")
2
3  weight = int(input("กรณป้อนน้ำหนัก (กิโลกรัม): "))
4  height = int(input("กรณป้อนส่วนสูง (เซนติเมตร): "))
5  age = int(input("กรณป้อนอายุ (ปี): "))
6  sex = input("เลือกเพศ (ชาย/หญิง): ").lower()
7
8  if sex == "ชาย":
9      bmr = 10 * weight + 6.25 * height - 5 * age + 5
10     print(f"อัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR): {bmr:.2f}")
11 elif sex == "หญิง":
12     bmr = 10 * weight + 6.25 * height - 5 * age - 161
13     print(f"อัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR): {bmr:.2f}")
14
```

c:/bmr.py

คำนวณอัตราการเผาผลาญแคลอรี

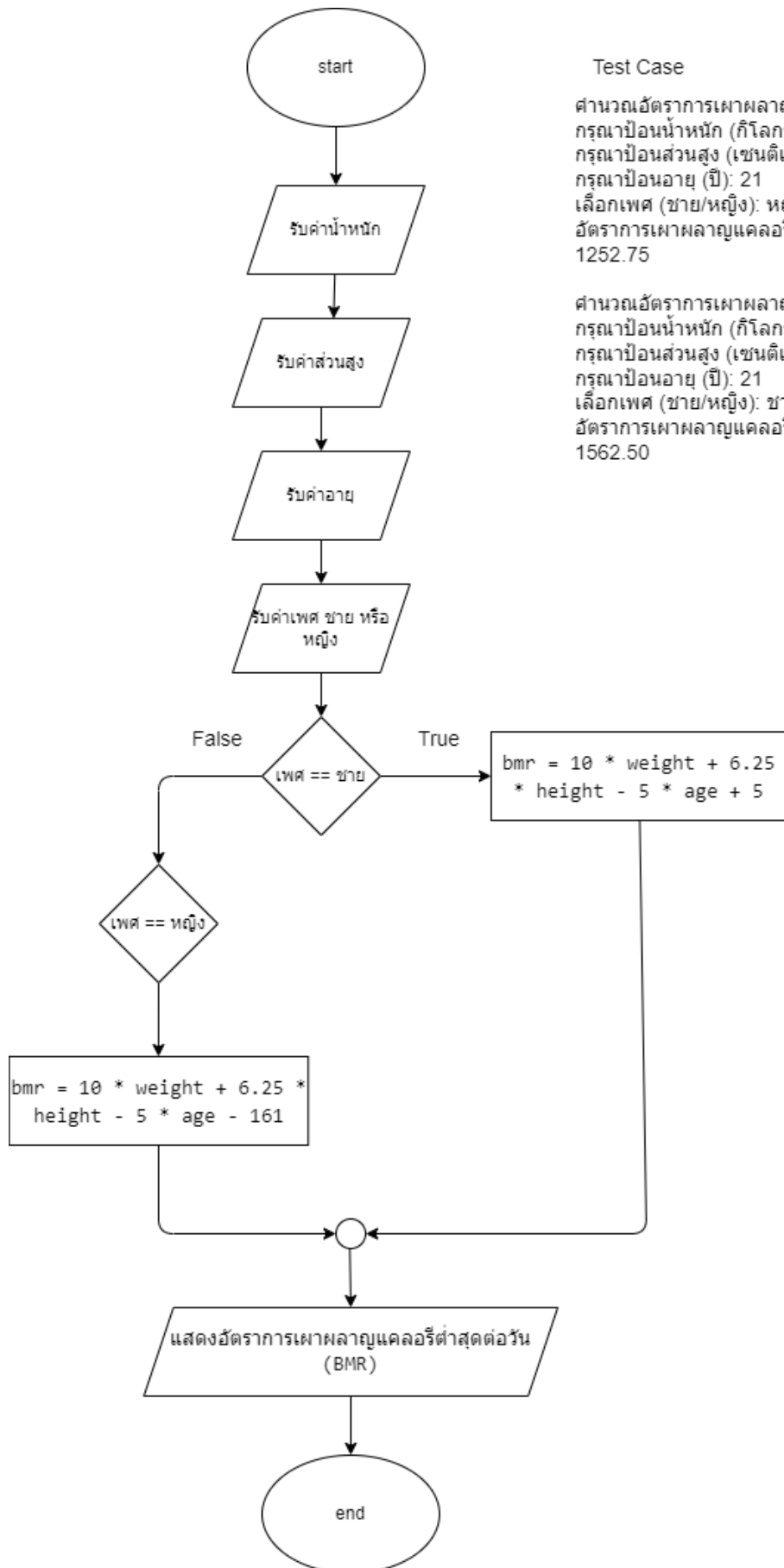
กรณป้อนน้ำหนัก (กิโลกรัม): 55

กรณป้อนส่วนสูง (เซนติเมตร): 155

กรณป้อนอายุ (ปี): 21

เลือกเพศ (ชาย/หญิง): หญิง

อัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR): 1252.75



Test Case

คำนวณอัตราการเผาผลาญแคลอรี
 กรณูป้อนน้ำหนัก (กิโลกรัม): 55
 กรณูป้อนส่วนสูง (เซนติเมตร): 155
 กรณูป้อนอายุ (ปี): 21
 เลือกเพศ (ชาย/หญิง): หญิง
 อัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR)
 1252.75

คำนวณอัตราการเผาผลาญแคลอรี
 กรณูป้อนน้ำหนัก (กิโลกรัม): 60
 กรณูป้อนส่วนสูง (เซนติเมตร): 170
 กรณูป้อนอายุ (ปี): 21
 เลือกเพศ (ชาย/หญิง): ชาย
 อัตราการเผาผลาญแคลอรีต่ำสุดต่อวัน (BMR)
 1562.50

โปรแกรมที่หาร 3 ลงตัว

week3 / 3.py / start

```
1 start = int(input("ป้อนตัวเลขเริ่มต้น: "))
2 end = int(input("ป้อนตัวเลขเริ่มต้น: "))
3
4 print("ตัวเลขที่หาร 3 ลงตัวได้แก่: ")
5
6 for num in range(start + 1, end):
7     if num % 3 == 0:
8         print(num)
9
10
```

PS C:\Users\ravip\OneDrive\เดสก์ท็อป\Pro

ป้อนตัวเลขเริ่มต้น: 3

ป้อนตัวเลขเริ่มต้น: 15

ตัวเลขที่หาร 3 ลงตัวได้แก่:

6

9

12

PS C:\Users\ravip\OneDrive\เดสก์ท็อป\Pro

คำนวณสี่เหลี่ยมผืนผ้า

week3 > cal_perimeter.py > L

```
1 L = float(input("Enter L: "))
2 W = float(input("Enter W: "))
3 P = 2 * (L + W)
4 print(f"Perimeter: {P}")
```

perimeter.py


Enter L: 30

Enter W: 50

Perimeter: 160.0

PS C:\Users\ravip\OneDr

คำนวณเงินกู้พร้อมคิดดอกเบี้ย

week3 >  cal_total_amount.py > [๑] lone

```
1 lone = float(input("ป้อนจำนวนเงินกู้(บาท): "))
2
3 if lone <= 1000:
4     interest = 0.10
5 elif lone <= 10000:
6     interest = 0.05
7 else:
8     interest = 0.02
9
10 total = lone * (1 + interest)
11
12 print("จำนวนเงินที่ต้องคืนรวมดอกเบี้ย: {:.2f} บาท".format(total))
```

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\ProblemSolving> & C:
total_amount.py
ป้อนจำนวนเงินกู้(บาท): 3000
จำนวนเงินที่ต้องคืนรวมดอกเบี้ย: 3150.00 บาท
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\ProblemSolving> █
```

หาเลขที่คำนวณลงตัว

week3 >  test2.py > ...

```
1 while True:
2     start = int(input("ป้อนตัวเลขเริ่มต้น: "))
3     end = int(input("ป้อนตัวเลขสิ้นสุด: "))
4     divisor = int(input("ป้อนตัวเลขที่ต้องการหารลงตัว: "))
5
6     print(f"\nตัวเลขที่อยู่ระหว่าง {start} ถึง {end} ที่หาร {divisor} ลงตัว ได้แก่:")
7     for num in range(start + 1, end):
8         if num % divisor == 0:
9             print(num)
10
11     stop = input("\nพิมพ์ 0 เพื่อหยุดการทำงาน หรือกด Enter เพื่อทำงานต่อ: ")
12     if stop == "0":
13         print("จบการทำงาน")
14         break
```

z.py

ป้อนตัวเลขเริ่มต้น: 1

ป้อนตัวเลขสิ้นสุด: 5

ป้อนตัวเลขที่ต้องการหารลงตัว: 2

ตัวเลขที่อยู่ระหว่าง 1 ถึง 5 ที่หาร 2 ลงตัว ได้แก่:

2

4

พิมพ์ 0 เพื่อหยุดการทำงาน หรือกด Enter เพื่อทำงานต่อ: █

คำนวณ GPA

Week2 > GPA.py > ...

```
1 def calculate_gpa(grades):
2     """
3     ฟังก์ชันคำนวณเกรดเฉลี่ยแบบมีหน่วยกิต
4     :param grades: รายการที่เก็บคู่ของ (เกรด, หน่วยกิต) [(grade, credit)]
5     :return: เกรดเฉลี่ย (GPA) (float)
6     """
7     total_points = 0
8     total_credits = 0
9
10    for grade, credit in grades:
11        total_credits += credit
12        total_points += grade_to_point(grade) * credit
13
14    if total_credits == 0:
15        return 0 # ถ้าไม่มีหน่วยกิตเลย
16
17    return total_points / total_credits
18
19 def grade_to_point(grade):
20     """
21     แปลงเกรดเป็นค่าคะแนน (Grade Points)
22     :param grade: เกรด (A, B, C, D, F)
23     :return: ค่าคะแนน (4.0, 3.0, 2.0, 1.0, 0.0)
24     """
25     grade_mapping = {
26         "A": 4.0,
27         "B": 3.0,
28         "C": 2.0,
29         "D": 1.0,
30         "F": 0.0
31     }
32     return grade_mapping.get(grade.upper(), 0)
33
```

```

33
34 # รับข้อมูลจากผู้ใช้
35 grades = []
36 print("กรุณาใส่เกรดและหน่วยกิตของแต่ละวิชา (พิมพ์ 'done' เมื่อเสร็จ):")
37
38 while True:
39     grade = input("ใส่เกรด (A, B, C, D, F): ").upper()
40     if grade == "DONE":
41         break
42
43     if grade not in ["A", "B", "C", "D", "F"]:
44         print("กรุณาใส่เกรดที่ถูกต้อง (A, B, C, D, F)")
45         continue
46
47     try:
48         credit = float(input("ใส่หน่วยกิต: "))
49         if credit <= 0:
50             print("หน่วยกิตต้องมากกว่า 0")
51             continue
52         grades.append((grade, credit))
53     except ValueError:
54         print("กรุณาใส่ตัวเลขสำหรับหน่วยกิต")
55
56 # คำนวณเกรดเฉลี่ย
57 gpa = calculate_gpa(grades)
58 print(f"เกรดเฉลี่ย (GPA): {gpa:.2f}")
59

```

.....

กรุณาใส่ เกรดและหน่วยกิตของแต่ละวิชา (พิมพ์ 'done' เมื่อเสร็จ):

ใส่ เกรด (A, B, C, D, F): A

ใส่หน่วยกิต: 3

กรุณาใส่ เกรดและหน่วยกิตของแต่ละวิชา (พิมพ์ 'done' เมื่อเสร็จ):

ใส่ เกรด (A, B, C, D, F): A

ใส่หน่วยกิต: 3

ใส่ เกรด (A, B, C, D, F): A

ใส่หน่วยกิต: 3

ใส่หน่วยกิต: 3

ใส่ เกรด (A, B, C, D, F): B

ใส่หน่วยกิต: 3

ใส่ เกรด (A, B, C, D, F): done

ใส่หน่วยกิต: 3

ใส่ เกรด (A, B, C, D, F): done

เกรดเฉลี่ย (GPA): 3.50

หาพื้นที่ของสามเหลี่ยม

Week2 > TriangleArea.py > ...

```
1  # เริ่มต้นการคำนวณพื้นที่สามเหลี่ยม 10 รูป
2  print("โปรแกรมคำนวณพื้นที่สามเหลี่ยมจำนวน 10 รูป")
3
4  # ตั้งค่าตัวนับเริ่มต้น
5  counter = 1
6
7  # วงรอบเพื่อคำนวณพื้นที่สามเหลี่ยม 10 รูป
8  while counter <= 10:
9      print(f"\nรูปที่ {counter}")
10
11     # รับค่าฐานและความสูงจากผู้ใช้
12     x = float(input("กรอกความยาวฐานของสามเหลี่ยม (X): "))
13     y = float(input("กรอกความสูงของสามเหลี่ยม (Y): "))
14
15     # คำนวณพื้นที่
16     area = (x * y) / 2
17
18     # แสดงผลลัพธ์
19     print(f"พื้นที่ของสามเหลี่ยมคือ: {area:.2f}")
20
21     # เพิ่มค่าตัวนับ
22     counter += 1
23
24 print("\nจบการทำงาน")
25
```

ng1eAm ea.py

โปรแกรมคำนวณพื้นที่สามเหลี่ยมจำนวน 10 รูป

รูปที่ 1

กรอกความยาวฐานของสามเหลี่ยม (X): 30

กรอกความสูงของสามเหลี่ยม (Y): 80

พื้นที่ของสามเหลี่ยมคือ: 1200.00

รูปที่ 2

กรอกความยาวฐานของสามเหลี่ยม (X): █

BIG O

week4 > slicing.py > ...

```
1 def reverse_slicing(s):
2     return s[::-1]
3
4 input_str = 'INE-KMUTNB'
5
6 if __name__ == "__main__":
7     print('Reverse String using recursive =', reverse_slicing(input_str))
8
```

slicing.py

Reverse String using recursive = BNTUMK-ENI

บวก

Test-week1 > test1.py > ...

```
1 print("***Calculate sum of two number***")
2
3 first_number = int(input("Enter the first number: "))
4 second_number = int(input("Enter the second number: "))
5
6 sum = first_number + second_number
7
8 print(f"The sum is: {sum} ")
```

/test1.py

Calculate sum of two number

Enter the first number: 1

Enter the second number: 2

The sum is: 3

ลบ

Test-week1 > test2.py > ...

```
1 print("***Calculate subtraction of two number***")
2
3 first_number = int(input("Enter the first number: "))
4 second_number = int(input("Enter the second number: "))
5
6 subtraction = first_number - second_number
7
8 print(f"The sum is: {subtraction} ")
9
```

```
/test2.py
***Calculate subtraction of two number***
Enter the first number: 2
Enter the second number: 9
The sum is: -7
```

หาเลขคี่ คู่

```
Test-week1 > test3.py > ...
1  print("***Check even or odd numbers***")
2
3  number = int(input("Enter a number: "))
4
5  if number % 2 == 0:
6      print("the number even.")
7  else:
8      print("the number odd.")
9
/test3.py
***Check even or odd numbers***
Enter a number: 3
the number odd.
```

หาฟาเรนไฮต์จากเซลเซียส

```
Test-week1 > test4.py > ...
1  print("***Convert Celsius to Fahrenheit***")
2
3  celsius = float(input("Enter temperture in Celsius: "))
4
5  fahrenheit = (celsius * 9/5) + 32
6
7  print(f"Temperture in Fahrenheit: {fahrenheit: .2f}")
8
```

```

/test4.py
***Convert Celsius to Fahrenheit***
Enter temperture in Celsius: 30
Temperture in Fahrenheit: 86.00

```

หา bmi

```

Test-week1 > test5.py > ...
1  print("***Convert BMI***")
2
3  # รับข้อมูลน้ำหนักและส่วนสูง
4  weight = float(input("Enter your weight (kg): "))
5  height = float(input("Enter your height (cm): "))
6
7  # แปลงความสูงจากเซนติเมตรเป็นเมตร
8  height = height / 100
9
10 # คำนวณ BMI
11 bmi = weight / (height ** 2)
12
13 # แสดงผลลัพธ์ BMI
14 print(f"Your BMI is {bmi:.2f}")
15

```

```

/test5.py
***Convert BMI***
Enter your weight (kg): 55
Enter your height (cm): 155
Your BMI is 22.89

```

คำนวณเลขเริ่มถึงเลขสิ้นสุด เช่น 5 - 8 ก็ 5+6+7+8

```

Test-week1 > test6.py > ...
1  print("***Calculate the sum between start and stop number***")
2
3  start = int(input("Enter the start number: "))
4  end = int(input("Enter the end number: "))
5
6  total_sum = sum(range(start, end + 1))
7
8  print(f"The sum from {start} to {end} is {total_sum}")
9

```

```

7 test7.py
***Calculate the sum between start and stop number***
Enter the start number: 5
Enter the end number: 8
The sum from 5 to 8 is 26
D:\C:\Users\pawin\OneDrive\Documents\ProblemSolving\

```

เกมทายเลข

Test-week1 > test7.py > ...

```

1  import random
2
3  print("***Welcome to the Number Gussing Game!***")
4  print("I'm thinking of a number between 1 and 100. Can you guss it?")
5
6  target_number = random.randint(1, 100)
7
8  attempts = 0
9
10 while True:
11     guess = int(input("Enter your guess: "))
12     attempts += 1
13
14     if guess < target_number:
15         print("Too low! Try again.")
16     elif guess > target_number:
17         print("Too high! Try again.")
18     else:
19         print(f"Congratulation! Yo guessed the number in {attempts} attempts.")
20         break

```

```

Too low! Try again.
Enter your guess: 60
Too low! Try again.
Enter your guess: 70
Too high! Try again.
Enter your guess: 65
Congratulation! Yo guessed the number in 8 attempts.

```

โจทย์: โปรแกรมคำนวณเงินฝากของจำนวน N บัญชี (Savings Account)

1. ให้เขียน

- Problem analysis chart
- Structure chart or interactivity chart
- IPO chart (ให้แยกการทำงานออกเป็น module ย่อยๆ)
- Algorithm
- Flowchart
- Pseudocode

2. ให้เขียนโปรแกรมตาม Flowchart ที่ได้ออกแบบไว้

ตัวอย่างการทำงานของโปรแกรม

Input number of Saving Account: 2

Input Principal of SA 1 : 10000

Input Principal of SA 2 : 5000

Input Interest of SA 1 : 0.5

Input Interest of SA 2 : 0.25

```
test1.py > ...
1  print("*** โปรแกรมคำนวณดอกเบี้ยบัญชีออมทรัพย์ ***")
2
3  # รับจำนวนบัญชี
4  จำนวนบัญชี = int(input("จำนวนบัญชีออมทรัพย์: "))
5
6  # ตัวแปรสำหรับสะสมดอกเบี้ยรวม
7  ดอกเบี้ยรวม = 0
8
9  # วงลูปคำนวณแต่ละบัญชี
10 for ลำดับ in range(1, จำนวนบัญชี + 1):
11     เงินต้น = float(input(f"ป้อนเงินต้นของบัญชีที่ {ลำดับ} (บาท): "))
12     อัตราดอกเบี้ย = float(input(f"ป้อนอัตราดอกเบี้ยของบัญชีที่ {ลำดับ} (ตัวอย่าง: 0.05 สำหรับ 5%): "))
13     ดอกเบี้ย = เงินต้น * อัตราดอกเบี้ย
14     ดอกเบี้ยรวม += ดอกเบี้ย
15     print(f"ดอกเบี้ยของบัญชีที่ {ลำดับ}: {ดอกเบี้ย:.2f} บาท\n")
16
17 # แสดงผลรวม
18 print(f"ดอกเบี้ยรวมจากทุกบัญชี: {ดอกเบี้ยรวม:.2f} บาท")
19
```

*** โปรแกรมคำนวณดอกเบี้ยบัญชีออมทรัพย์ ***

จำนวนบัญชีออมทรัพย์: 2

ป้อนเงินต้นของบัญชีที่ 1 (บาท): 10000

ป้อนอัตราดอกเบี้ยของบัญชีที่ 1 (ตัวอย่าง: 0.05 สำหรับ 5%): 0.5

ดอกเบี้ยของบัญชีที่ 1: 5000.00 บาท

ป้อนเงินต้นของบัญชีที่ 2 (บาท): 5000

ป้อนอัตราดอกเบี้ยของบัญชีที่ 2 (ตัวอย่าง: 0.05 สำหรับ 5%): 0.25

ดอกเบี้ยของบัญชีที่ 2: 1250.00 บาท

ดอกเบี้ยรวมจากทุกบัญชี: 6250.00 บาท

PS C:\Users\rawip\OneDrive\เดสก์ท็อป\ProblemSolving> █