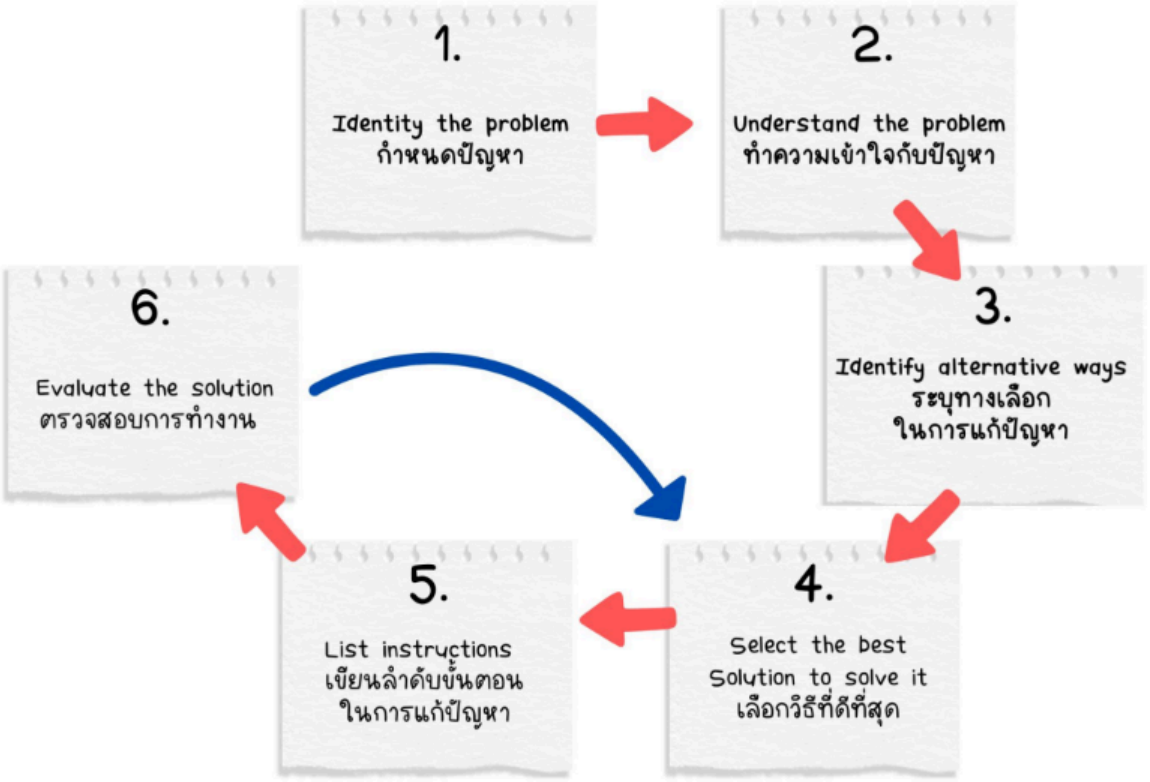
นายพีรกานต์ ไกรพินิจ 6806022511129

Problem Solving with Computer Homework 1-1

ใช้วิธีการแก้ปัญหา 6 ขั้นตอน



1.) Identity the problem การกำหนดปัญหา

โจทย์ปัญหาคือ การเขียนโปรแกรมทอนเงิน โดยมีกระบวนการของการจ่ายเงิน และมีการทอนเงินกลับไปโดยมีเงื่อนไขในการตรวจสอบ แบงค์ธนบัตรเช่น แบงค์ 1000 , 500 , 100 , 50 , 20 และ เหรียญ 10 , 5 , 2 , 1 ว่ามีอยู่เท่าไรและต้องทอนเงินโดยการใช้แบงค์หรือเหรียญประเภทใดจึงจะสามารถทอนเงินได้ครบ

2.) Understand the problem

จะมี input อะไรบ้าง โดยที่มีการรับ input เข้ามาในส่วนของราคาสินค้าที่ขาย และจำนวนเงินที่จ่ายมา

มี output อะไรบ้าง โดยมีการแสดง output ออกมาเป็นจำนวนเงินที่ทอนไป โดยมีการระบุว่า จ่ายด้วยแบงค์อะไรเป็นจำนวนเท่าไร เช่น input ที่รับเข้ามา(ลูกค้าจ่ายเงิน) และตัว output จะแสดงเป็นจำนวนเงิน(ตัวเลข) และระบุแบงค์ที่ทอนกลับไป(ระบุชนิดของแบงค์พร้อมจำนวนแบงค์ที่ทอนกลับไป) ยกตัวอย่าง ราคาสินค้าราคา 1290 บาท และมีการจ่ายเงินมาเป็นจำนวน 1500 บาท จากนั้นจะแสดงเป็นจำนวนเงินทอนคือ 210 บาท พร้อมระบุว่าเป็นแบงค์ 100 บาท จำนวน 2 ใบ และ เหรียญ 10 บาท จำนวน 1 เหรียญ

โดยมีข้อกำหนดว่าเมื่อมีการจ่ายเงินมา จะต้องมาตรวจสอบแล้วว่า จำนวนเงิน พร้อมประเภทของแบงค์ธนบัตร + เหรียญบาท เหลือเท่าใด โดยมีการตรวจสอบเป็นลำดับตั้งแต่ แบงค์ 1000 บาท ที่เป็นจำนวนมูลค่าสูงสุด เป็นอันดับแรก และเรียงลำดับการตรวจสอบลงมาเป็นขั้นตอน จนมาถึงเหรียญ 1 บาท ยกตัวอย่างเช่น ราคาสินค้า 1290 บาท แล้วมีการจ่ายเงินสำหรับซื้อสินค้ามาเป็นจำนวน 1500 บาท จะมีการตรวจสอบว่าจำนวนเงินที่ต้องทอนเป็นจำนวนกี่บาท จากนั้นให้นำเงินที่ต้องทอนนั้นไปตรวจสอบว่า ต้องจ่ายด้วยแบงค์ธนบัตรชนิดใดเป็นจำนวนเท่าไร โดยยกตัวอย่างสมมติว่า ต้องทอนเงินกลับไปเป็นจำนวน 210 บาท เมื่อตรวจสอบเงื่อนไขแล้วว่าแบงค์ธนบัตร 100 บัตร มีจำนวนที่ไม่พอหรือหมดจะต้องทอนด้วยธนบัตรชนิดใดเป็นลำดับถัดไป และ ต้องใช้จำนวนเท่าไร เช่น ธนบัตร 100 บาท มีอยู่ 1 ใบ จะทำการทอนด้วย ธนบัตร 100 บาท 1 ใบ และไปตรวจสอบเงื่อนไขที่เหลืออย่าง แบงค์ธนบัตร 50 บาท ว่ามีอยู่หรือไม่หากมีให้ทอนเป็นจำนวนเท่าใดต่อไป

3.) Identify Alternative Ways

ใช้ if-else Statement ในการตรวจสอบเงื่อนไขว่ามีแบงค์ธนบัตรแต่ละชนิดอยู่เท่าใด และตรวจสอบว่าต้องจ่ายเป็นเงินทอนกลับไปเป็นจำนวนเท่าใดจึงเหมาะสม

4.) Select the Best Solution

ต้องใช้วิธีใดในการคิดคำนวณและลดจำนวนการทำงานของโปรแกรมให้ใช้ทรัพยากรและเวลาให้มีประสิทธิภาพที่สุดในการทำงานของโปรแกรมทอนเงิน เช่นมีการเขียนโค้ดดักทางที่อาจ

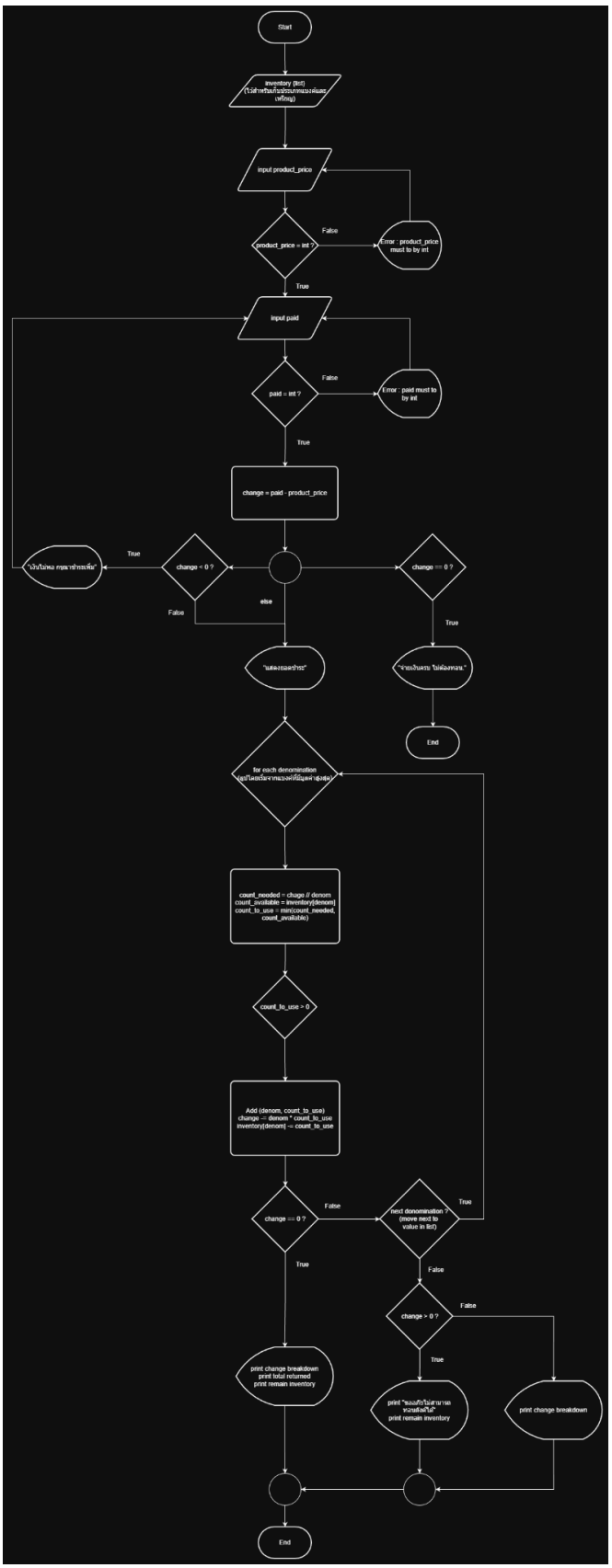
จะเกิดบัคของโปรแกรมเพื่อไม่ให้โปรแกรมมีการทำงานที่อาจทำให้เกิดความผิดพลาด ลดการทำงานของโปรแกรมที่ซ้ำซ้อนเพื่อให้ประสิทธิภาพโปรแกรมดีขึ้น

5.) List Instructions

เขียน Algorithm

โดยขั้นตอนแรกจะต้องมีการกำหนดค่าต่าง ๆ ทั้งมีการเก็บยอดแบงค์ธนบัตรและเหรียญ ว่ามีเท่าไร ไว้ใช้สำหรับการทำ Test Case ต่าง ๆ และมีการป้อน input ของราคาสินค้า และ จำนวนเงินที่จะใช้จ่ายในการชำระสินค้า จากนั้นทำการคำนวณราคาสินค้าลบกับจำนวนเงินที่ต้องชำระ และมีการสร้างเงื่อนไขสำหรับโปรแกรมทอนเงินที่ต้องมีการตรวจสอบว่าจะต้องทอนด้วยแบงค์ธนบัตรและเหรียญใดบ้าง และจำนวนเท่าไร รวมเป็นทั้งหมดเท่าไร

เขียน Flowchart



เขียน Program

```
inventory = {
    1000: 2,
    500: 5,
    100: 10,
    50: 10,
    20: 20,
    10: 30,
    5: 20,
    2: 50,
    1: 0
}

DENOMS = sorted(inventory.keys(), reverse=True)

def get_int_input(prompt):
    while True:
        try:
            value = int(input(prompt))
            if value < 0:
                print("Error: value cannot be negative.")
                continue
            return value
        except ValueError:
            print("Error: Please enter a valid integer.")

def calculate_change(change, inventory):
```

```
    result = []

    for denom in DENOMS:
        if change == 0:
            break

        count_needed = change // denom
        count_available = inventory[denom]
        count_to_use = min(count_needed, count_available)

        if count_to_use > 0:
            result.append((denom, count_to_use))
            change -= denom * count_to_use
            inventory[denom] -= count_to_use

    # change เหลือเท่าไร นั่นคือยอดที่ทอนไม่ได้
    return result, change

def print_breakdown(change_list):
    print("\nรายละเอียดการทอนเงิน:")
    total = 0
    for denom, count in change_list:
        subtotal = denom * count
        total += subtotal
        print(f"{denom} x {count} = {subtotal}")
    print("จำนวนเงินที่ทอนไป:", total)
```

```
product_price = get_int_input("Enter product price: ")

while True:
    paid = get_int_input("Enter paid amount: ")
    change = paid - product_price

    if change < 0:
        print(f"ยอดชำระไม่พอ ต้องการเพิ่มอีก {product_price - paid} บาท.")
    else:
        break

if change == 0:
    print("ชำระครบจำนวน ไม่ต้องทอน.")
else:
    print(f"\nจำนวนที่ต้องทอน = {change}")

    result, remaining = calculate_change(change, inventory)

    if remaining > 0:
        print("\n!! สต็อกเหรียญ/ธนบัตรไม่เพียงพอ ทอนไม่ครบ !!")
        print(f"ยอดคงเหลือที่ไม่ได้ทอน: {remaining}")

    print_breakdown(result)
```

6.) Evaluate the solution

ตรวจสอบ Program (Debugging)

ต้องมีการตรวจสอบ Debugging อย่างเวลาป้อน input ต้องมีการเขียนโค้ดดักสำหรับ input เป็นตัวเลขต้องเขียนดักในกรณีที่ถ้าป้อน string เข้าไปต้องแจ้ง Error ถ้าใส่ตัวเลขติดลบ ต้องแจ้ง Error ถ้าจำนวนเงินในการทอน หรือ ชำระเงินไม่พอต้อง แจ้งว่าอะไร

ตรวจสอบผลลัพธ์ (Result Validation)

โดยในโปรแกรมที่ทางผู้จัดได้ทำขึ้นมาจะมี Test Case อยู่ทั้งหมด 5 เคส คือ

6.1) ชำระเงินปกติ

6.2) ป้อน input ผิด (ทั้ง ราคาสินค้า และ และจำนวนเงินที่ชำระ)

6.3) ทดสอบชำระเงินไม่พอ จะแสดงอย่างไร

6.4) ทดสอบในกรณีที่ ถ้าธนบัตร / เหรียญมีไม่พอจะทำงานอย่างไรหรือแสดงอย่างไร

6.5) ชำระเงินแบบพอดี