Digital Wallet

d66_f1_digital_wallet

อธิบายโจทย์

- รัฐบาลมีนโยบายกระตุ้นเศรษฐกิจโดยการแจกเงิน เราจะต้องเขียนคลาสชื่อ Digital Wallet เพื่อเป็นระบบกลางสำหรับรัฐและประชาชนในการดำเนินนโยบายนี้
- ประชาชนจะต้องติดตั้ง app เพื่อรับเงิน โดยรัฐบาลจะแจกเงินให้กับคนต่างๆ เป็นจำนวนเงินที่ต่างกัน ที่เวลาต่างกัน และระยะเวลาในการใช้งานเงินนี้ต่างกัน เมื่อหมดระยะเวลาก็จะไม่สามารถใช้งานได้ การแจกเงินต้องสามารถทำซ้ำได้กับประชาชน คนเดิม
- แต่ละคนในประเทศสามารถระบุได้โดยใช้ “รหัสประชาชน” เป็น string ความยาวไม่เกิน 10 ตัวอักษร
- คลาสนี้จะต้องมีฟังก์ชันการทำงานดังนี้

  add_money , use_money , current_money , status

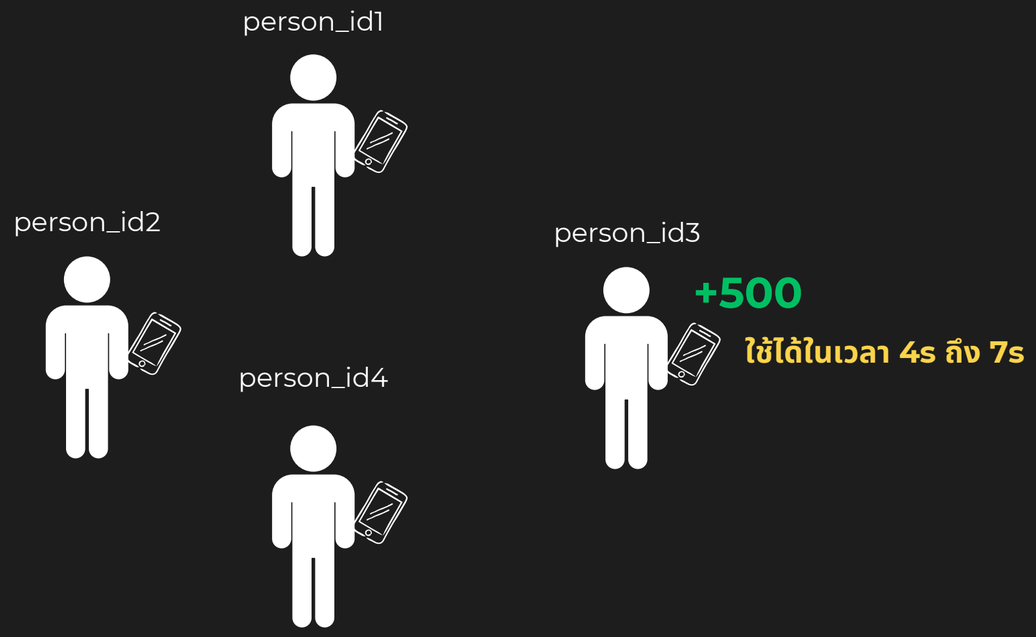
อธิบายโจทย์

```cpp
#ifndef __STUDENT_H_
#define __STUDENT_H_

#include <string>

// you can include anything

using namespace std;

class DigitalWallet {
  // you can declare variables or write new function

 public:
  void add_money(size_t time, string person_id, int amount, size_t duration) {
    // your code here

  }

  bool use_money(size_t time, string person_id, int amount) {
    // your code here

  }

  int current_money(size_t time, string person_id) {
    // your code here

  }

  void status(size_t time, long long &total_give, long long &total_spent,
              long long &total_expired) {
    // your code here

  }
};

#endif
```

อธิบายโจทย์

```cpp
void add_money(size_t time, string person_id, int amount, size_t duration) {
  // your code here

}
```

ณ เวลา time มีการ เพิ่มเงินจำนวน amount ให้กับ wallet ของคนที่มีid คือ person_id โดยเงินดังกล่าวจะสามารถใช้ได้ จนถึงเวลา time + duration เท่านั้น

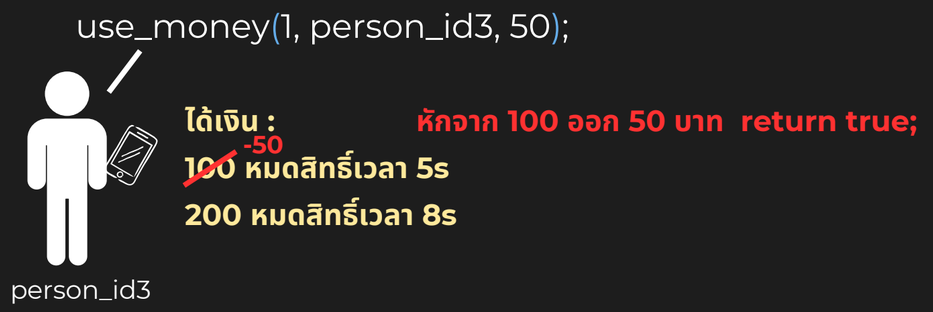


อธิบายโจทย์

```
bool use_money(size_t time, string person_id, int amount) {
  // your code here

}
```

ณ เวลา time คนที่มี id คือ person_id ต้องการใช้เงินจำนวน amount จาก wallet โดยเงินที่ใช้นี้จะถูกหักออกจากเงินที่อยู่ใน wallet ของคนดังกล่าว

- ฟังก์ชันนี้จะต้องคืนค่า true หากคนดังกล่าวมีเงินใน wallet ที่ยังใช้ได้เพียงพอต่อการใช้งาน และเงินใน wallet จะถูกหักออกไปตามจำนวน amount แต่ถ้าหากจำนวนเงินที่ยังใช้ได้ไม่พอ เงินจะไม่ถูกหัก และฟังก์ชันนี้จะต้องคืนค่า false
- การหักเงินนั้น จะหักจากเงินที่พ้นระยะเวลาการใช้งานเร็วกว่าก่อนเสมอ (เช่น หากเราได้เงิน มาสองก้อน คือ 100 กับ 200 โดยที่ก้อน 100 นั้นพ้นระยะเวลาการใช้งานก่อนก้อน 200 แล้ว เราจะใช้เงิน 50 บาท เราจะหักเงินจากก้อน 100 ก่อน)



อธิบายโจทย์

```
int current_money(size_t time, string person_id) {
  // your code here

}
```

คืนจำนวนเงินที่ยังใช้ได้ของคนที่มี id เป็น person_id ณ เวลา time
(ไม่รวมเงินที่พ้นระยะเวลาใช้งานไปแล้ว และไม่รวมเงินที่ถูกใช้ไปแล้ว)

current_money(10, person_id3);

person_id3

ได้เงิน :
300 ใช้หมดแล้ว
100 หมดสิทธิ์เวลา 5s
200 หมดสิทธิ์เวลา 15s

return 200;

อธิบายโจทย์

```
void status(size_t time, long long &total_give, long long &total_spent,
            long long &total_expired) {
  // your code here

}
```

คำนวณสถานะจำนวนเงินรวมของทั้งระบบ ณ เวลา time เมื่อเรียกฟังก์ชันนี้ จะต้องตั้งค่าให้กับตัวแปรต่างๆ ดังนี้

- total_give คือจำนวนเงินรวมทั้งหมดที่รัฐบาลได้แจกมา
- total_spent คือจำนวนเงินรวมทั้งหมดที่ถูกใช้ผ่านฟังก์ชัน use_money ที่คืนค่าเป็น true
- total_expired คือจำนวนเงิน รวมทั้งหมดที่หมดอายุไปก่อนโดยไม่ได้ใช้งาน

แนวทางการแก้โจทย์ปัญหานี้(ภาพรวมโดยคร่าวๆ)

เริ่มด้วย Constraint ของข้อนี้

รับประกันว่าในการใช้งานคลาสนี้ สถานการณ์ต่าง ๆ ต่อไปนี้จะเป็นจริงเสมอ

- การเรียกฟังก์ชันใด ๆ ที่มีค่า time นั้น รับประกันว่าการเรียกแต่ละครั้งค่า time จะไม่น้อยลงเลย
- สำหรับการเรียก add_money แต่ละครั้งนั้น รับประกันว่าเงินที่ add_money มาในการเรียกครั้งล่าสุดจะพ้นระยะเวลาการใช้งาน “ไม่เร็วกว่า” เงินจากการ add_money ที่ถูกเรียกมาก่อนหน้า

สมมติฐานเกี่ยวกับการใช้งาน

ฟังก์ชันทั้ง 4 นั้นสามารถถูกเรียกใช้งานเป็นจำนวนแตกต่างกันอย่างไรก็ได้ ไม่จำเป็นว่าต้องมีการเรียกฟังก์ชันหนึ่งมากกว่าอีกฟังก์ชันหนึ่ง และความซับซ้อนเชิงเวลาที่คาดหวังไว้ของคลาสนี้คือ โดยเฉลี่ยแล้ว หากมีการเรียกฟังก์ชันทั้งหมด k ครั้ง เวลาที่ใช้ควรจะเป็น O(k)

หมายความว่า method ต่างๆ จะต้อง amortized O(1)

ชุดข้อมูลทดสอบ

สำหรับชุดทดสอบใด ๆ จะมี จำนวนคำสั่ง <= 1,000,000 คำสั่ง และมีจำนวนคนในระบบ <= 50,000 คน และค่าเวลาที่เรียกใช้งานฟังก์ชันจะไม่เกิน 1,000,000,000

- 10% จำนวนคำสั่ง <= 100 มีคนในระบบเพียงคนเดียว และมีแต่คำสั่ง add_money กับ current_money
- 10% จำนวนคำสั่ง <= 100 และมีแต่คำสั่ง add_money กับ current_money
- 15% มีคนในระบบเพียงคนเดียว และมีแต่คำสั่ง add_money กับ current_money
- 20% มีคนในระบบเพียงคนเดียว
- 45% ไม่มีเงื่อนไขอื่น ๆ

เขียนไปแค่ 2 method นี้ก่อน ก็ได้คะแนนบางส่วนแล้ว

- **35 / 100 คะแนน**

ทำแค่ 2 method ตามนี้ก่อน

Data Structure ที่ใช้

```
unordered_map<string, queue<pair<int, int>>> toAddQ;
unordered_map<string, int> currentMoney;
long long total_give = 0, total_spent = 0, total_expired = 0;
```

person_id → queue ของ {เงินที่จะเพิ่มมา, เวลาหมดอายุ}
person_id → เงินรวมของคนนั้นๆ ณ ปัจจุบัน

Method ต่างๆ

method เขียนเอง

```
void updateMoneyOnTime(size_t time, string person_id){
  queue<pair<int, int>> & q = toAddQ[person_id];
  int expiredMoney = 0;
  while (!q.empty() && q.front().second < time) {
    expiredMoney += q.front().first, q.pop();
  }
  currentMoney[person_id] -= expiredMoney;
  total_expired += expiredMoney; // global status
}
```

method นี้มีเพื่อ update เงินปัจจุบันของ คนนั้นๆ (person_id)

1. คำนวณเงินที่จะต้องหมดอายุไป และ update เงินของคนนั้นๆด้วย
2. update ค่าของ total_expired

```
void add_money(size_t time, string person_id,
               int amount, size_t duration) {
  updateMoneyOnTime(time, person_id);
  toAddQ[person_id].emplace(amount, time + duration);
  currentMoney[person_id] += amount;
  total_give += amount;
}
```

```
int current_money(size_t time, string person_id) {
  updateMoneyOnTime(time, person_id);
  return currentMoney[person_id];
}
```

update เงินของ คนนั้นๆ ก่อนเสมอ (person_id)

- **55 / 100 คะแนน**
  ทำเพิ่มขึ้นอีก 2 method

Data Structure ที่ใช้

```
unordered_map<string, queue<pair<int, int>>> toAddQ;
unordered_map<string, int> currentMoney;
long long total_give = 0, total_spent = 0, total_expired = 0;
```

person_id → queue ของ {เงินที่จะเพิ่มมา, เวลาหมดอายุ}
person_id → เงินรวมของคนนั้นๆ ณ ปัจจุบัน

Method ต่างๆ

```
void updateMoneyOnTime(size_t time, string person_id)
```

method นี้มีเพื่อ update เงินปัจจุบันของ คนนั้นๆ (person_id)
(รายละเอียดอยู่หน้าก่อนนี้)

```
bool use_money(size_t time, string person_id, int amount) {
  updateMoneyOnTime(time, person_id);
  if (currentMoney[person_id] < amount) return false;
  currentMoney[person_id] -= amount;
  total_spent += amount; // global status

  queue<pair<int, int>> & q = toAddQ[person_id];
  while (!q.empty() && amount ≥ q.front().first) {
    amount -= q.front().first; q.pop();
  }
  if (!q.empty()) q.front().first -= amount;
  return true;
}
```

update เงินให้เป็นของเวลานั้นก่อน

update หัวคิวให้ลบเงินที่ใช้แล้วออกไปด้วย
เพื่อให้ updateMoneyOnTime update เงินถูกต้องในครั้งถัดๆไป

```
int expiredMoney = 0;
while (!q.empty() && q.front().second < time) {
  expiredMoney += q.front().first, q.pop();
}
currentMoney[person_id] -= expiredMoney;
```

(updateMoneyOnTime)

```
void status(size_t time, long long &total_give,
                         long long &total_spent,
                         long long &total_expired) {
  total_give = this→total_give;
  total_spent = this→total_spent;
  string personId;
  for (auto & p : toAddQ) {
    personId = p.first;
    updateMoneyOnTime(time, personId);
  }
  total_expired = this→total_expired;
}
```

update เงินให้เป็นของเวลานั้นสำหรับคนทุกคน ซึ่งจะทำให้มี method นี้
ไม่ amortized O(1) เพราะจะต้อง loop หมดบนคนทุกคน
เดี๋ยวเราจะปรับให้ดีขึ้นในหน้าถัดไป

• 100 / 100 คะแนน

Data Structure ที่ใช้

```
unordered_map<string, queue<pair<int, int>>> toAddQ;
unordered_map<string, int> currentMoney;
long long total_give = 0, total_spent = 0, total_expired = 0;
```

person_id → queue ของ {เงินที่จะเพิ่มมา, เวลาหมดอายุ}
person_id → เงินรวมของคนนั้นๆ ณ ปัจจุบัน

Method ต่างๆ

```
void updateMoneyOnTime(size_t time, string person_id)
```

method นี้มีเพื่อ update เงินปัจจุบันของ คนนั้นๆ (person_id)
(รายละเอียดอยู่หน้าก่อนนี้)

```
queue<pair<int, string>> expireQ;
```

Q ของ {เวลาหมดอายุ, person_id}

✓ เพิ่ม queue นี้มาเป็น field ช่วย เป็น queue ที่เก็บเวลาที่ expire → person_id
ใช้เป็นเหมือน lookup table เพื่อจะได้ไม่ต้อง loop บนคนทุกคน

```
void status(size_t time, long long &total_give,
                         long long &total_spent,
                         long long &total_expired) {
  total_give = this->total_give;
  total_spent = this->total_spent;
  string personId;
  while (!expireQ.empty() && expireQ.front().first < time) {
    personId = expireQ.front().second; expireQ.pop();
    updateMoneyOnTime(time, personId);
  }
  total_expired = this->total_expired;
}
```

ไล่เช็คจาก expiredQ แทนที่จะ loop บน person_id

```
void add_money(size_t time, string person_id,
               int amount, size_t duration) {
  updateMoneyOnTime(time, person_id);
  toAddQ[person_id].emplace(amount, time + duration);
  currentMoney[person_id] += amount;
  total_give += amount;
  expireQ.emplace(time + duration, person_id);
}
```

(อย่าลืมไปแก้ใน add_money ด้วยนะครับ hehe)

[d66_f1_digital_wallet] Digital Wallet

Limit: 1.0s, 512MB

Read | File

```cpp
#ifndef __STUDENT_H_
#define __STUDENT_H_

#include <string>
#include <unordered_map>
#include <queue>

using namespace std;

class DigitalWallet {

  unordered_map<string, queue<pair<int, int>>> toAddQ;
  unordered_map<string, int> currentMoney;
  queue<pair<int, string>> expireQ;

  void updateMoneyOnTime(size_t time, string person_id){
    queue<pair<int, int>> & q = toAddQ[person_id];
    int expiredMoney = 0;
    while (!q.empty() && q.front().second < time) {
      expiredMoney += q.front().first, q.pop();
    }
    currentMoney[person_id] -= expiredMoney;
    total_expired += expiredMoney; // global status
  }

  public:
  int total_give = 0, total_spent = 0, total_expired = 0;

  void add_money(size_t time, string person_id,
                 int amount, size_t duration) {
    updateMoneyOnTime(time, person_id);
    toAddQ[person_id].emplace(amount, time + duration);
    currentMoney[person_id] += amount;
    total_give += amount;
    expireQ.emplace(time + duration, person_id);
  }


  bool use_money(size_t time, string person_id, int amount) {
    updateMoneyOnTime(time, person_id);
```

Latest Submission Status Refresh

839472 (12 tries) compiler msg

less than a minute ac (15:17:38)

100.0 [PPPPPP PP]

Helper Refres [What's this?]

-- There is no comment for this submission --

AI help by `gemini-2.5-pro` Get

AI help by `Claude-3.5-Sonnet` Get

สมาชิก

นายกิตติเชษฐ์ อารยะสุจินต์	6732005321
นายจิรภัทร ไชยยา	6732007621
นายปิติวรรธณ์ กิตติวิมลชัย	6732023621
นายพศิน ธัญญกสิกล	6732025921

THANK YOU