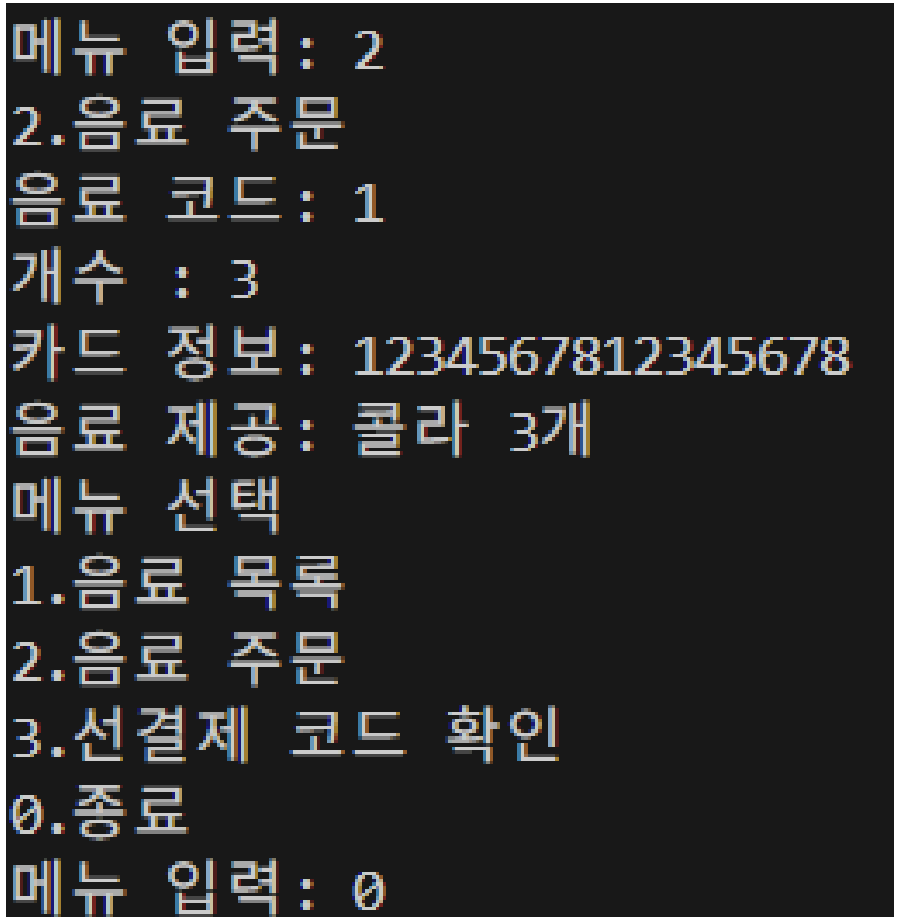
2nd Cycle Presentation

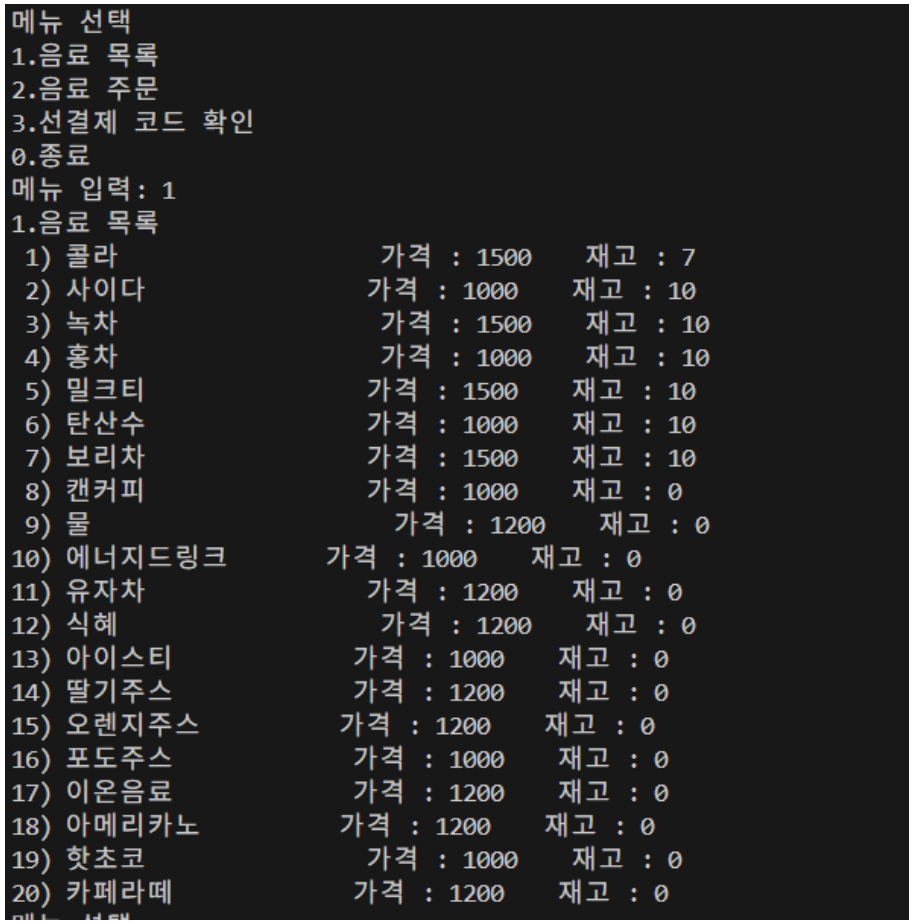
5조

목차

- 구현
- 테스트 결과
- Resolution 여부

구현





https://drive.google.com/file/d/1ZkeLzXd9Z7pPAON8263qrCSKRwY7c7iz/view?usp=sharing

실행 방법

src > network > NetworkManager.cpp > NetworkManager()

```
17  NetworkManager::NetworkManager() {
18      // addresses.emplace(1, Address{"127.0.0.1", 1111});
19      // addresses.emplace(2, Address{"127.0.0.1", 2020});
20      // addresses.emplace(3, Address{"127.0.0.1", 3030});
21      // addresses.emplace(4, Address{"127.0.0.1", 4040});
22      addresses.emplace(5, Address{"127.0.0.1", 5050});
23      addresses.emplace(6, Address{"127.0.0.1", 6060});
24      addresses.emplace(7, Address{"127.0.0.1", 7070});
25      itemManager = &ItemManager::getInstance();
26      prepaymentStock = &PrepaymentStock::getInstance();
27      messageFactory = &MessageFactory::getInstance();
28      dvmNavigator = nullptr;
29  }
```

- 다른 DVM의 정보를 미리 알고 있다고 했으므로 프로그램 상에서 다른 DVM의 IP 주소와 포트 넘버를 하드 코딩
- 테스트에서 3개까지만 사용하므로 나머지는 주석처리

실행 방법

build > {} T5.json > { } item_list > { } 3 > # price

```json
{
    "dvm_id": 5,
    "dvm_x": 0,
    "dvm_y": 0,
    "item_list": {
        "1": {
            "item_name": "콜라",
            "item_num": 10,
            "price": 1500
        },
        "2": {
            "item_name": "사이다",
            "item_num": 10,
            "price": 1000
        },
```

- json 파일로 각각의 DVM 좌표와 재고를 저장
- 해당 파일명을 프로그램 실행시 명령 인수로 전달하면 각각의 DVM을 실행할 수 있다.

```
root@906a82b0d1ac:/home/dev/build# ./MyApp.out T5.json
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력:
```

```
root@906a82b0d1ac:/home/dev/build# ./MyApp.out T6.json
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력:
```

카드 정보 규칙

카드사는 외부 시스템이므로 공란이 아닌지, 16자리 이상인지만 확인

테스트를 위해 카드사에 저장된 카드
- 잔액 충분한 카드 : 1234567812345678
- 잔액 없는 카드 : 0000000000000000

```
Bank::Bank() {
    cardBalanceMap["1234567812345678"] = 1000000000;
    cardBalanceMap["0000000000000000"] = 0;
}
```

테스트 실패 목록

Test Name	Description	Expected Output	Test Output
CP05	안 맞는 않은 개수 입력	에러 메시지 출력	다른 에러 메시지 출력
CP06	재고 없는 음료수 주문	에러 메시지 출력	기능 미구현
CP07	로컬에 없는 음료 주문	구매 취소	기능 미구현
CP08	부정확 카드 번호 입력	에러 메시지 출력	다른 에러 메시지 출력
CP09	잔고 부족 음료 주문	에러 메시지 출력	기능 미구현
CP10	잘못된 코드 입력	에러 메시지 출력	기능 미구현
CP11	정상 결제 가능값 입력	음료 정상 출고	카드 입력 미정의
CP12	정상 결제 가능값 입력	음료 정상 출고	카드 입력 미정의
CP13	정상 결제 가능값 입력	음료 정상 출고	카드 입력 미정의
CP14	정상 선결제 코드 입력	음료 정상 출고	기능 미구현
CP15	정상 선결제 코드 입력	음료 정상 출고	기능 미구현
CP16	정상 선결제 코드 입력	음료 정상 출고	기능 미구현

테스트 실패 목록

Test Name	Description	Expected Output	Test Output
BF05	매뉴 선택 시 숫자로 시작하는 문자열 입력	유효하지 않음	음료주문으로 넘어가고 무엇인가 유효하지 않은 것이 입력처리됨
BF09	음료 코드 입력 시 숫자로 시작하는 문자열 입력	유효하지 않음	개수 입력으로 넘어가고 유효하지 않은 것이 입력 처리됨
BF10	음료 코드 입력 시 범위 밖의 값 입력	유효하지 않음	개수 입력으로 진행
BF11	수량 입력 시 0~100 밖의 값 입력	유효하지 않음	카드 정보 입력으로 진행
BF12	수량 100개로 주문	유효하지 않음	"Connection Failed: Connection refused" 라는 메세지 출력 이후 예외처리

BF13이후 기능 미구현으로 테스트 불가

수정 결과 - CP05

범위 밖 코드 입력 시 다른 에러 메시지 출력

- 입력 속성에 대한 개별 체크를 통해 정상 에러 메시지 출력

```
std::unique_ptr<Payment> payment;
int itemCode, quantity;
while (true) {
    itemCode = readInt("음료 코드: ");
    payment = std::make_unique<Payment>(itemCode);
    if (payment->validate()) break;
    std::cout << "범위 밖 입력입니다.\n";
}
while (true) {
    quantity = readInt("개수 : ");
    payment = std::make_unique<Payment>(itemCode, quantity);
    if (payment->validate()) break;
    std::cout << "범위 밖 입력입니다.\n";
}
```

수정 결과 - CP06

범위 밖 수량 입력 시 다른 에러 메시지 출력

- 입력 속성에 대한 개별 체크를 통해 정상 에러 메시지 출력

```
std::unique_ptr<Payment> payment;
int itemCode, quantity;
while (true) {
    itemCode = readInt("음료 코드: ");
    payment = std::make_unique<Payment>(itemCode);
    if (payment->validate()) break;
    std::cout << "범위 밖 입력입니다.\n";
}
while (true) {
    quantity = readInt("개수 : ");
    payment = std::make_unique<Payment>(itemCode, quantity);
    if (payment->validate()) break;
    std::cout << "범위 밖 입력입니다.\n";
}
```

수정 결과- CP07

다른 DVM에만 재고가 있는 음료 구매 시도 후 선결제 취소

- 통신 문제 해결로 실행 가능

```
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력: 2
2.음료 주문
음료 코드: 10
개수 : 1
카드 정보: 1234567812345678
선결제 하시겠습니까?
1. 네
2. 아니오
입력:2
가장 가까운 자판기 좌표는 (3, 4)입니다.
```

수정 결과- CP08

카드 번호 입력 시 "[][][][][][][][][][][][]" 등 특수문자 포함 입력

- 카드 정보만 존재한다면 유효
- 카드 정보 없을 시 결제 실패 출력

```
메뉴 입력: 2
2.음료 주문
음료 코드: 1
개수 : 1
카드 정보: [][][][][][][][][][][][]
결제 실패
```

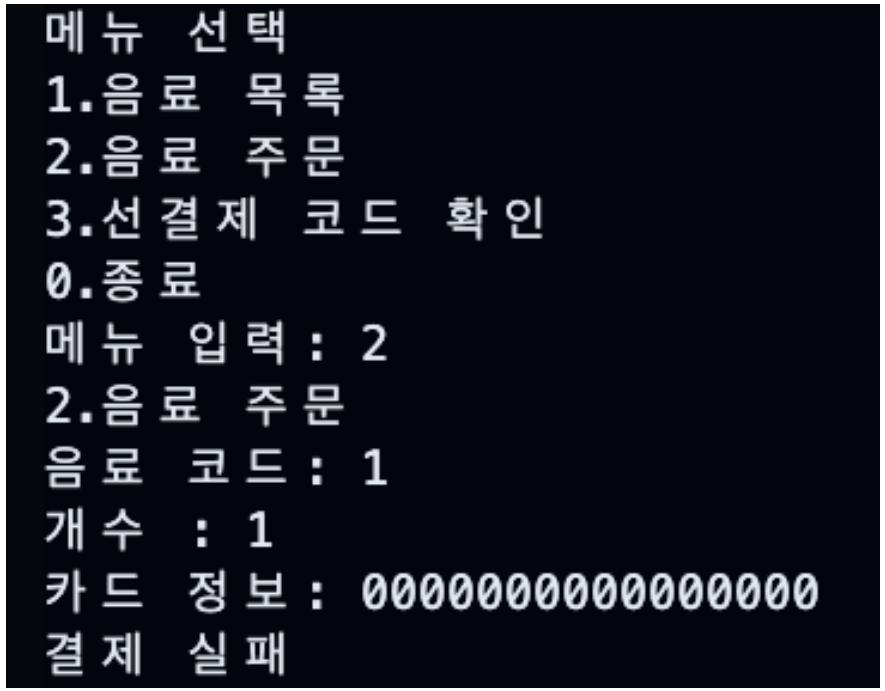
수정 결과- CP09

카드 잔고가 부족한 상태에서 결제 실행

- Test Output : 구매 기능 미 동작으로 해당 동작 테스트 불가능

이전 문서 부실로 제대로 된 테스트 불가능했음

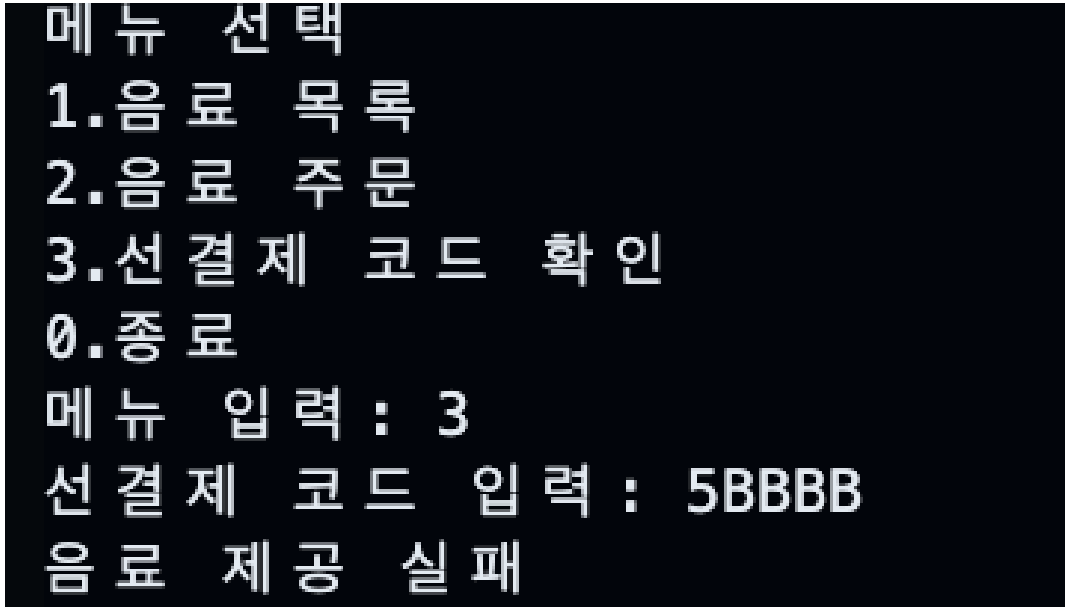
- 카드 잔고 부족시 결제 실패 출력



수정 결과- CP10

잘못된 선결제 코드 입력

- 통신 문제 해결로 실행 가능



수정 결과 - CP11

DVM 1대로 구매 시 미동작

- 통신 기능 구현 및 구매 기능 구현

```
bool Bank::pay(const Payment &payment) {
    const PaymentMethod *method = payment.getbuyContent();
    const CardPay *cardPay = dynamic_cast<const CardPay *>(method);

    if (cardPay) {
        std::string card = cardPay->getCardInfo();
        int amount = payment.getTotalPrice();

        if (!isValid(card)) {
            return false;
        }

        if (!hasEnoughBalance(card, amount)) {
            return false;
        }

        cardBalanceMap[card] -= amount;
        return true;
    }
```

수정 결과 - CP12

DVM 2대로 구매 시 미동작

- 통신 기능 구현 및 구매 기능 구현

```
bool Bank::pay(const Payment &payment) {
    const PaymentMethod *method = payment.getbuyContent();
    const CardPay *cardPay = dynamic_cast<const CardPay *>(method);

    if (cardPay) {
        std::string card = cardPay->getCardInfo();
        int amount = payment.getTotalPrice();

        if (!isValid(card)) {
            return false;
        }

        if (!hasEnoughBalance(card, amount)) {
            return false;
        }

        cardBalanceMap[card] -= amount;
        return true;
    }
```

수정 결과 - CP13

DVM 3대로 구매 시 미동작

- 통신 기능 구현 및 구매 기능 구현

```
bool Bank::pay(const Payment &payment) {
    const PaymentMethod *method = payment.getbuyContent();
    const CardPay *cardPay = dynamic_cast<const CardPay *>(method);

    if (cardPay) {
        std::string card = cardPay->getCardInfo();
        int amount = payment.getTotalPrice();

        if (!isValid(card)) {
            return false;
        }

        if (!hasEnoughBalance(card, amount)) {
            return false;
        }

        cardBalanceMap[card] -= amount;
        return true;
    }
```

수정 결과 - CP14

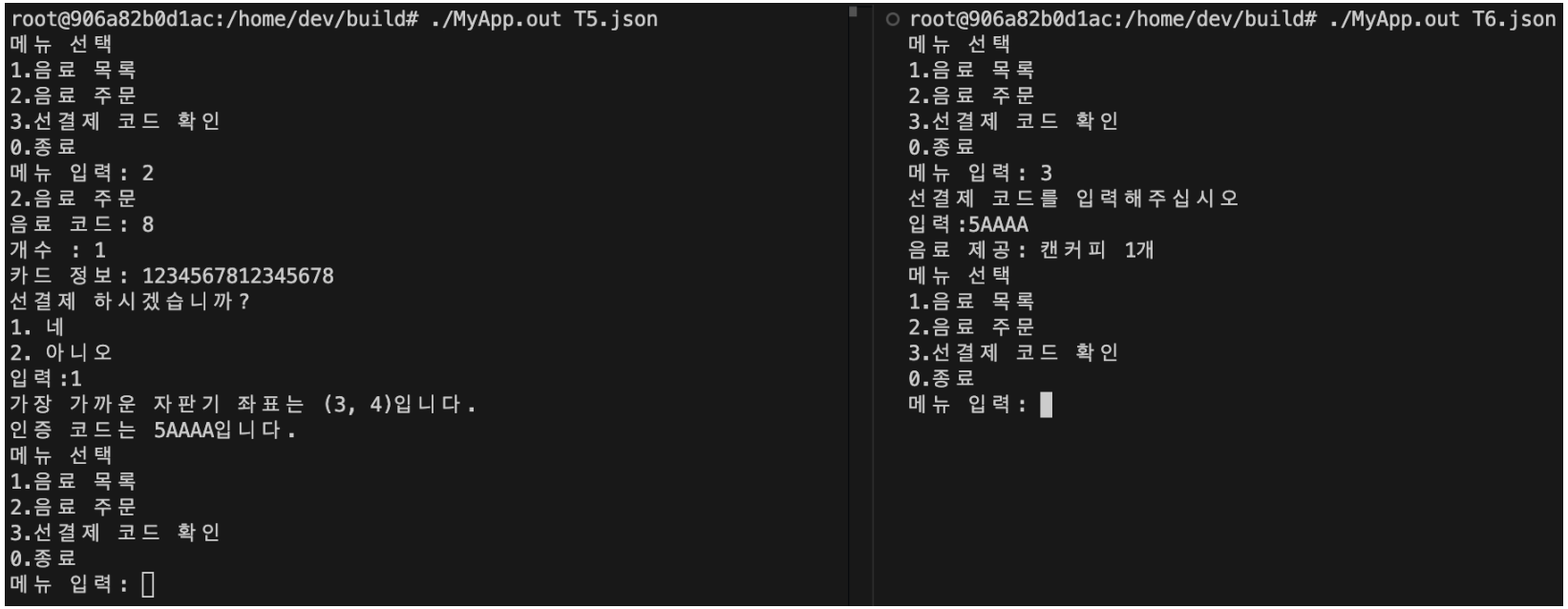
DVM 1대로 선결제 코드 입력시 미동작

- DVM 1대만 실행 시
인증코드를 발급 받을 수 없음

수정 결과 - CP15

DVM 2대로 선결제 코드 입력시 미동작

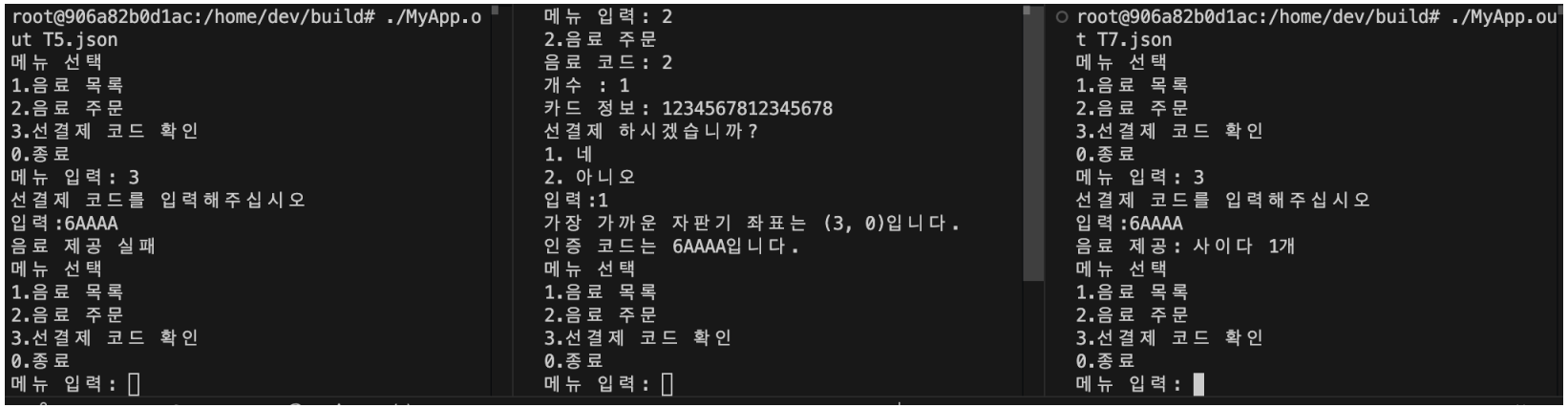
- 인증코드로 선결제한 음료 수령



수정 결과 - CP16

DVM 3대로 선결제 코드 입력시 미동작

- T5(0, 0), T6(3, 4), T7(3, 0)이고 2번 음료수는 양쪽 다 있으므로 T6에 더 가까운 T7 좌표 반환

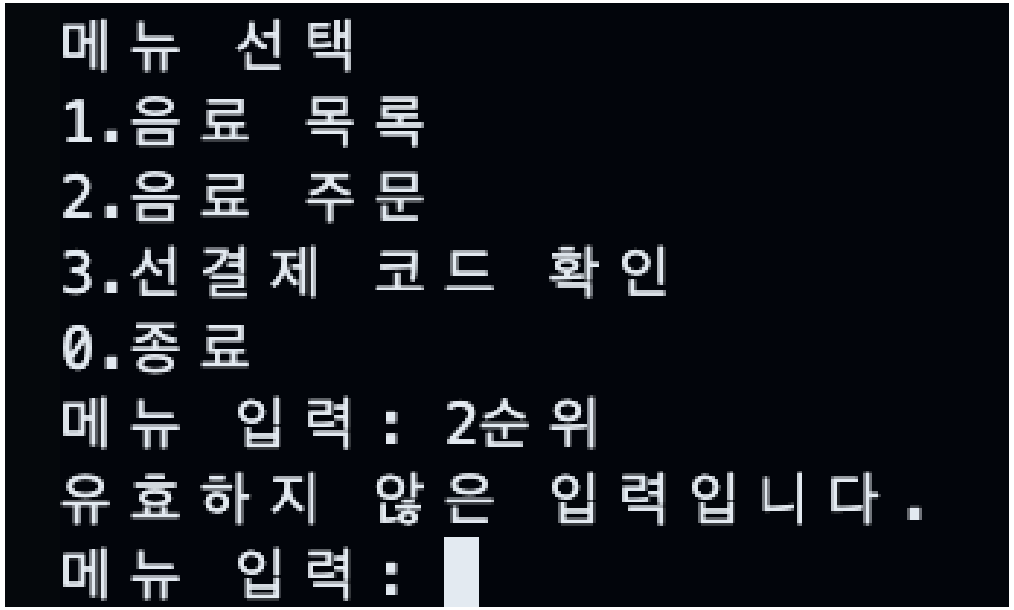


수정 결과 - BF05

매뉴 선택 시 숫자로 시작하는 문자열 입력

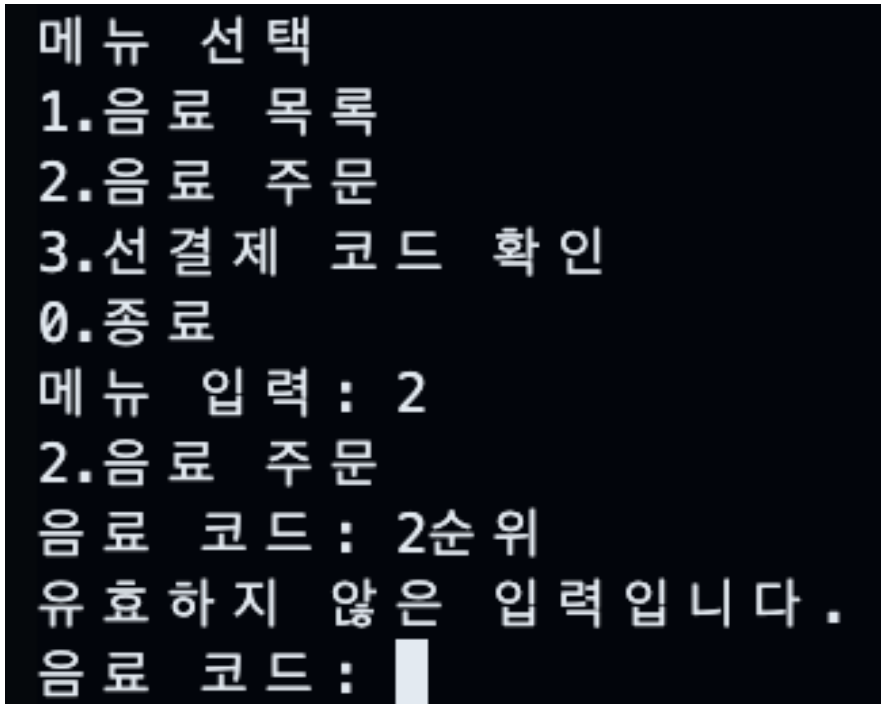
```cpp
int CLManager::readInt(std::string text) {
    while (true) {
        std::cout << text;
        std::string line;
        std::getline(std::cin >> std::ws, line); // 줄 전체 입력 (앞 공백 제거)

        std::istringstream iss(line);
        int value;
        std::string extra;
        if (iss >> value && !(iss >> extra)) {
            return value; // 오직 하나의 숫자만 있을 때
        }
        std::cout << "유효하지 않은 입력입니다.\n";
    }
}
```



수정 결과 - BF09

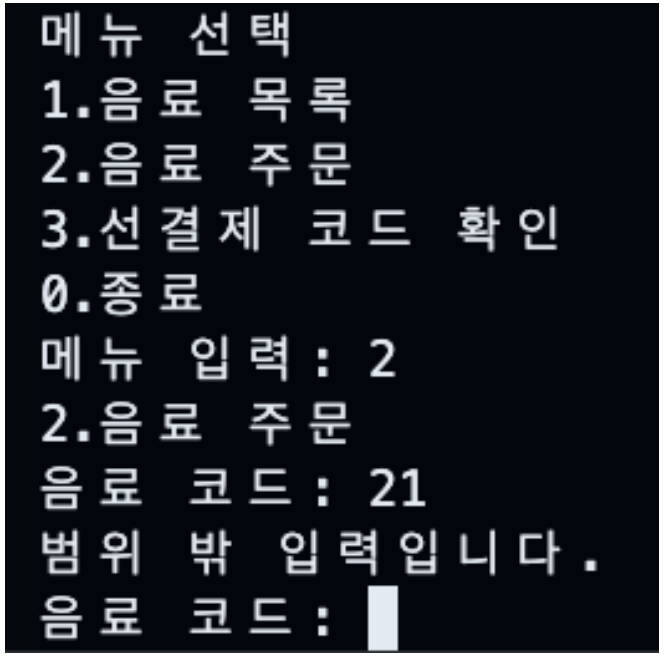
음료 코드 입력 시 숫자로 시작하는 문자열 입력



수정 결과 - BF10

음료 코드 입력 시 범위 밖의 값 입력

```
bool ItemManager::isValidType(std::optional<int
    if (itemcode < 1 || itemcode > 20){
        return false;
    }
```



수정 결과 - BF11

수량 입력 시 0~100 밖의 값 입력

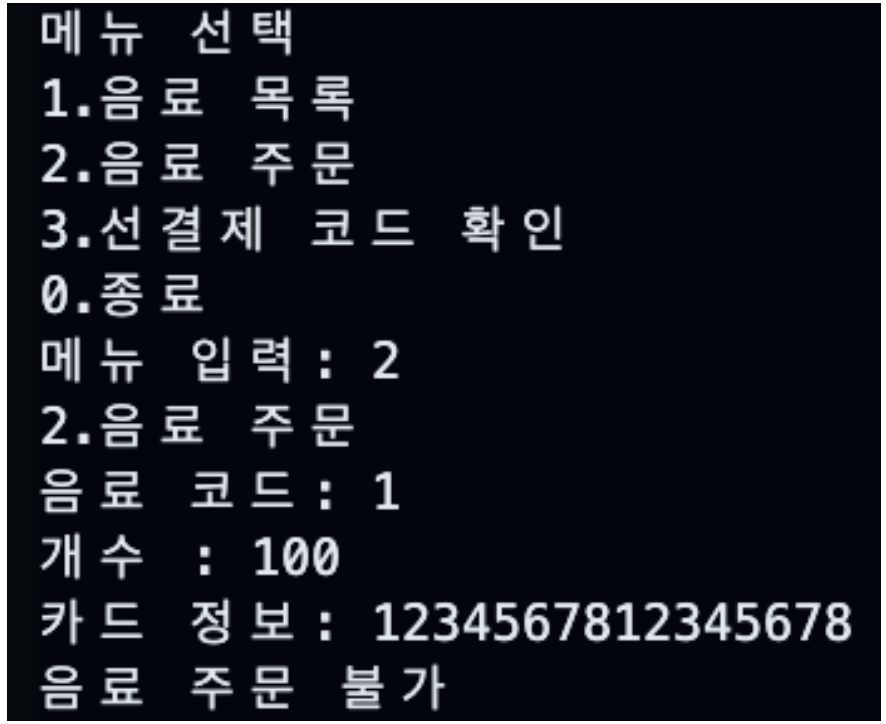
```
bool Item::isValidType(int reqCount) const{
    return reqCount >= 1 && reqCount <= 100;
}
```

```
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력 : 2
2.음료 주문
음료 코드 : 1
개수 : 999
범위 밖 입력입니다.
개수 :
```

수정 결과 - BF12

개수 100개 입력

- 모든 DVM에 음료 코드 1의 재고가 100개 미만이므로 음료 주문 불가 출력



수정 결과 - BF13

카드 정보에 - 를 포함해서 입력

- 카드사에 - 포함된 정보 저장되어 있다면 유효

```
bool CardPay::isValid() const {
    return !cardInfo.empty() && cardInfo.length() >= 16;
}
```

```
Bank::Bank() {
    cardBalanceMap["1234567812345678"] = 1000000000;
    cardBalanceMap["0000000000000000"] = 0;
}
```

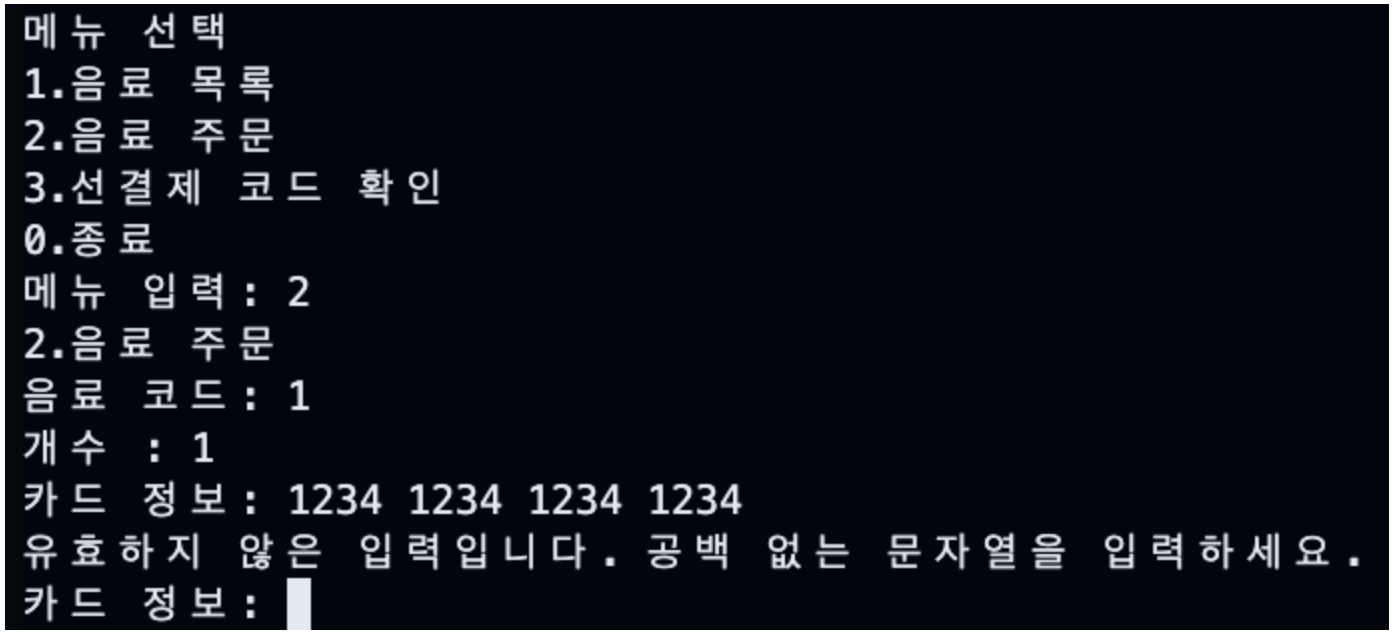
```
bool Bank::isValid(const std::string &cardInfo) const {
    return cardBalanceMap.find(cardInfo) != cardBalanceMap.end();
}

bool Bank::hasEnoughBalance(const std::string &cardInfo, int amount) const {
    auto it = cardBalanceMap.find(cardInfo);
    return it != cardBalanceMap.end() && it->second >= amount;
}
```

수정 결과 - BF14

카드 정보에 한글, 영문, - 이외의 특수문자, 공백 입력

- 카드사에 한글, 영문, - 이외의 특수문자정보 저장되어 있다면 유효
- 중간에 공백 입력시 유효하지 않은 입력으로 재입력 받음



수정 결과 - BF15

카드 정보에 공란 입력

```
bool CardPay::isValid() const {
    return !cardInfo.empty() && cardInfo.length() >= 16;
}
```

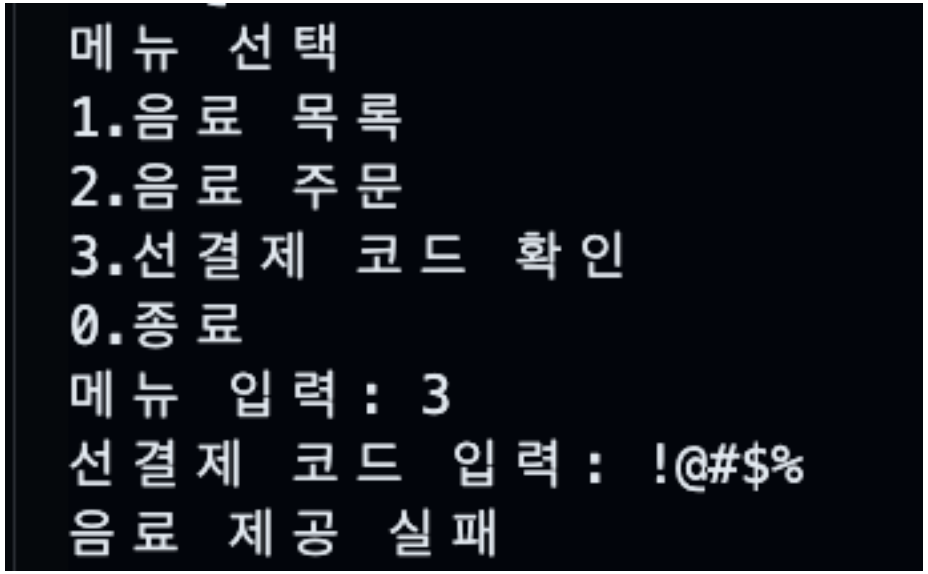
- 입력단 공란 입력 방지 및 카드 정보 공백 시 유효하지 않은 카드번호로 결제 실패

수정 결과 - BF16

인증코드에 특수문자, 공백 입력

- 입력단 공란 입력 방지
- 등록된 인증 코드 외의 모든 인증 코드

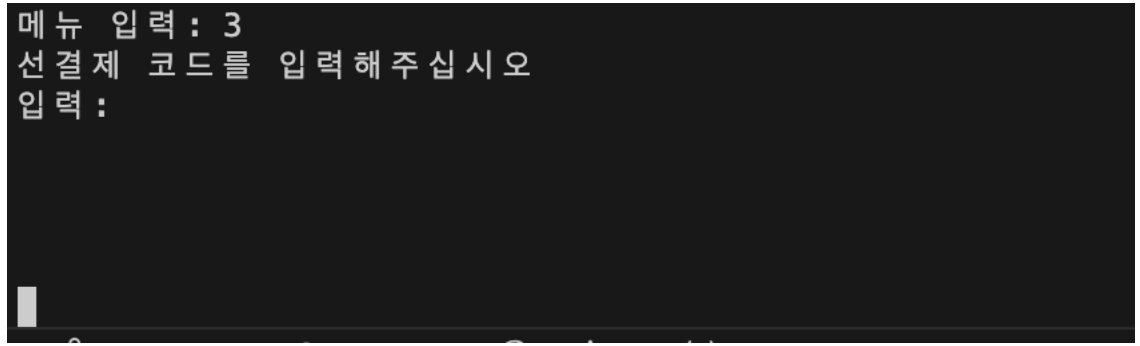
(특수 문자 포함된 인증코드도) 음료 제공 실패 출력



수정 결과 - BF17

인증코드에 공란 입력

- 코드 형식을 입력할 때까지 입력 대기



수정 결과 - BF18

이미 사용된 인증코드 재사용

- 이미 사용한 인증 코드는 재사용 불가능

```
메뉴 입력: 3
선결제 코드를 입력해주십시오
입력:6AAAA
음료 제공: 콜라 1개
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력: 3
선결제 코드를 입력해주십시오
입력:6AAAA
음료 제공 실패
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
```

수정 결과 - BF19

다른 VM에서 생성된 정상 인증코드 사용

- 선결제한 음료 제공

```
메뉴 입력: 3
선결제 코드를 입력해주십시오
입력:6AAAA
음료 제공: 콜라 1개
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
```

수정 결과 - BF20

선결제를 처리해준 VM이 아닌 별개의 VM에서 인증코드를 사용할 경우

- 재고를 확보한 VM에 인증코드를 등록하기 때문에 사용 불가

```
선결제 하시겠습니까?
1. 네
2. 아니오
입력:1
가장 가까운 자판기 좌표는 (0, 0)입니다.
인증 코드는 6AAAA입니다.
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력: 3
선결제 코드를 입력해주십시오
입력:6AAAA
음료 제공 실패
```

수정 결과 - BF21

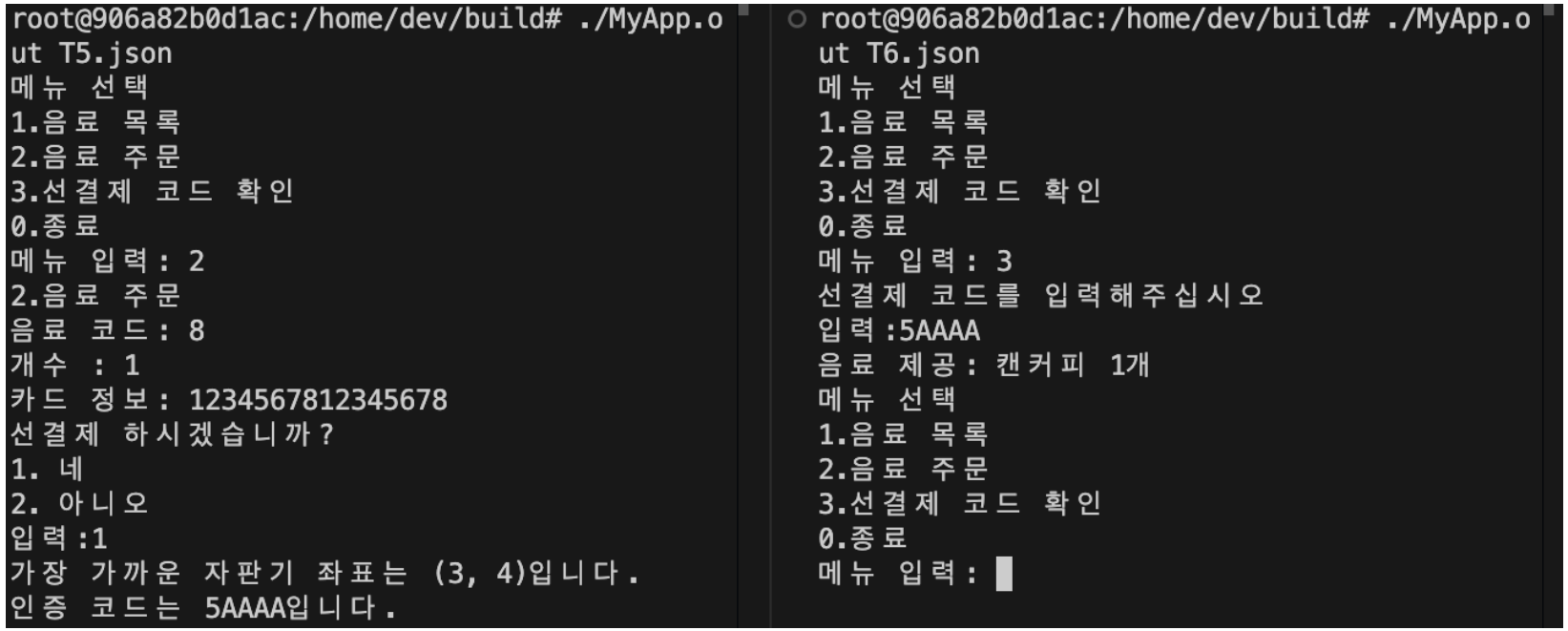
인증코드가 제한없이 무한히 입력 가능한지

- 정상 코드를 입력받을 때까지 반복문을 빠져나올 수 없었던 버그 수정

수정 결과 - BF24

선결제한 음료를 제공 받을 시 해당 음료 재고 정상 여부

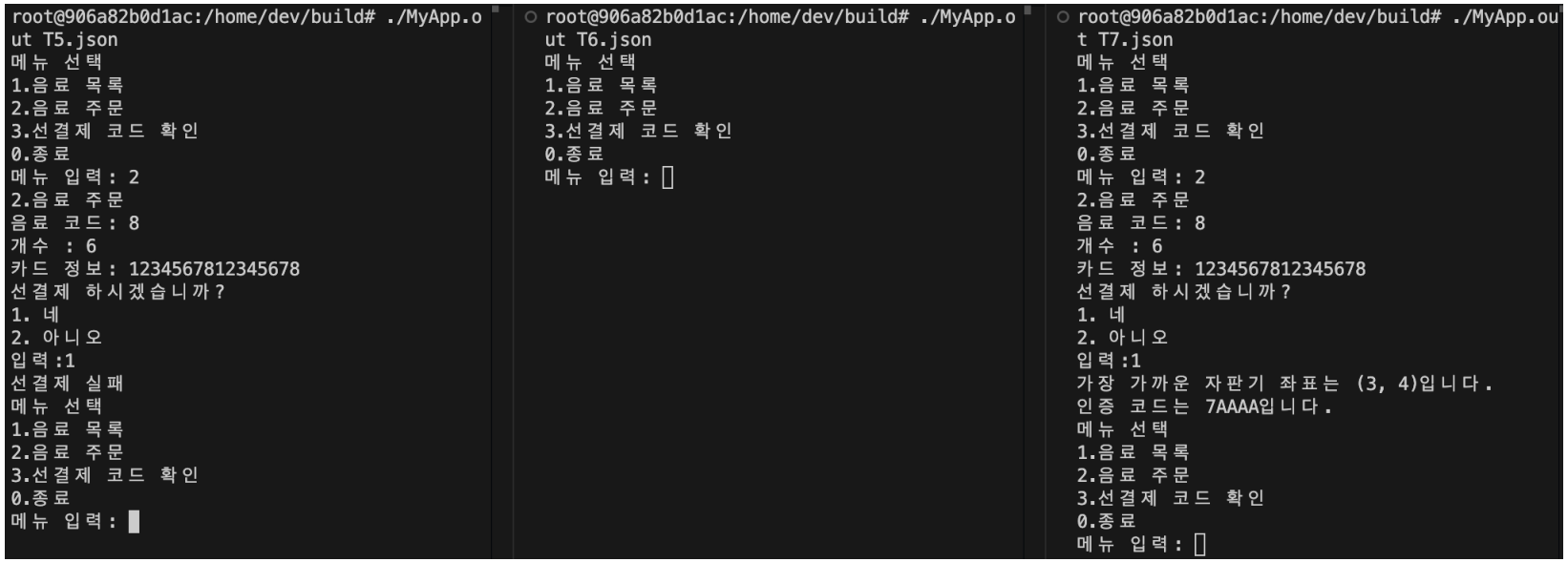
- 선결제한 대로 음료 제공



수정 결과 - BF25

동시에 2개 이상의 VM에서 선결제 요청이 오는 경우

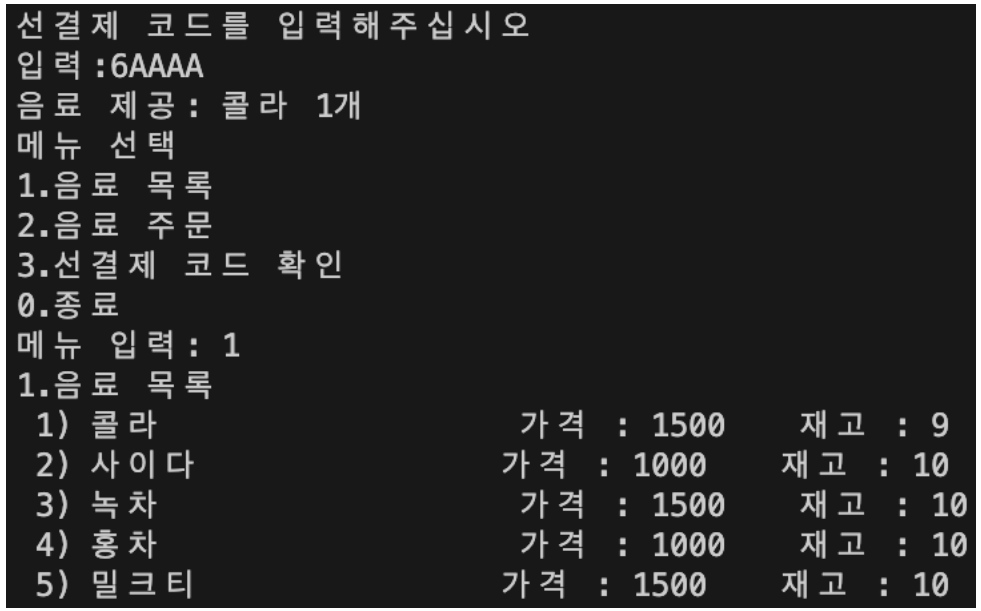
- 둘다 선결제 여부를 물어보고 먼저 응답한 VM에게 코드제공



수정 결과 - BF26

선결제된 물품의 재고가 정상적으로 감소되어 있는지

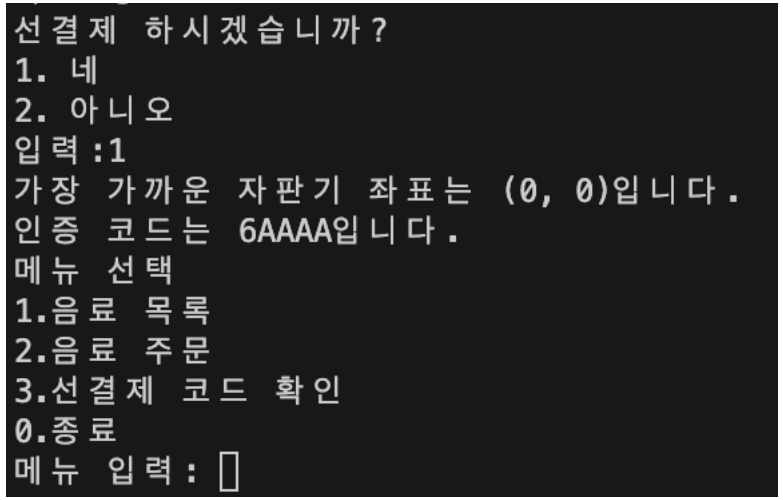
- 메모리 상에서 감소된 재고를 저장하고 있다가 프로그램이 정상 종료되면 json 파일에 반영



수정 결과 - BF27

선결제 완료 후 제공되는 정보(자판기 위치 정보 및 인증코드)가 정상적으로 표시되는지

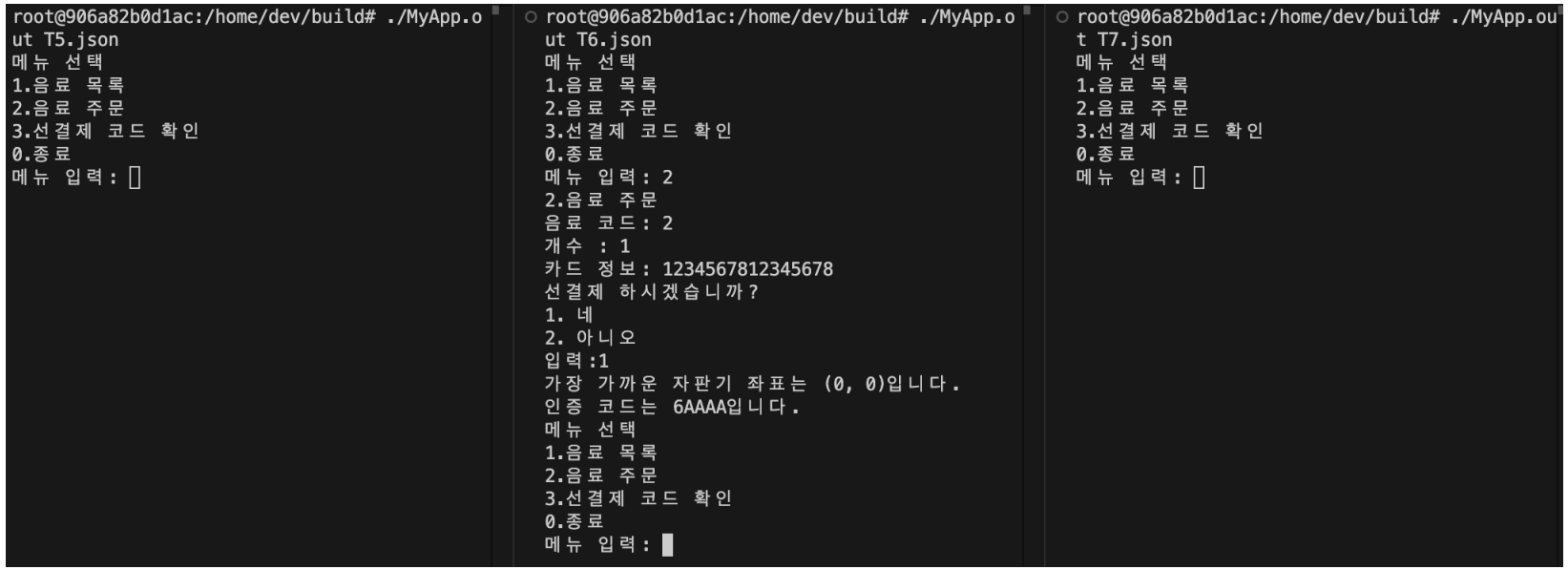
- 좌표와 인증 코드 출력



수정 결과 - BF28

같은 거리에 있는 2개 이상의 VM이 모두 선결제가 가능한 경우

- T5(0,0), T7(6,0) VM에 음료 2의 재고가 있고, T6(3,4)에서 선결제하면 둘 중 ID가 낮은 T5 VM의 좌표 안내



수정 결과 – BF30

재고가 있는 음료를 파악할 수 없음

- 1번 입력이 VM재고 출력

```
root@906a82b0d1ac:/home/dev/build# ./MyApp.out T7.json
메뉴 선택
1.음료 목록
2.음료 주문
3.선결제 코드 확인
0.종료
메뉴 입력: 1
1.음료 목록
 1) 콜라                    가격 : 1500    재고 : 0
 2) 사이다                  가격 : 1000    재고 : 9
 3) 녹차                    가격 : 1500    재고 : 0
 4) 홍차                    가격 : 1000    재고 : 0
 5) 밀크티                  가격 : 1500    재고 : 0
 6) 탄산수                  가격 : 1000    재고 : 0
 7) 보리차                  가격 : 1500    재고 : 0
 8) 캔커피                  가격 : 1000    재고 : 0
 9) 물                      가격 : 1200    재고 : 0
10) 에너지드링크          가격 : 1000    재고 : 0
11) 유자차                 가격 : 1200    재고 : 0
12) 식혜                   가격 : 1200    재고 : 0
13) 아이스티               가격 : 1000    재고 : 0
14) 딸기주스               가격 : 1200    재고 : 0
15) 오렌지주스             가격 : 1200    재고 : 10
16) 포도주스               가격 : 1000    재고 : 10
17) 이온음료               가격 : 1200    재고 : 10
18) 아메리카노             가격 : 1200    재고 : 10
19) 핫초코                 가격 : 1000    재고 : 10
20) 카페라떼               가격 : 1200    재고 : 10
```

BF22/BF23/BF29

- 프로그램 중간에 통신이 불가능한 경우 예외처리 부족..