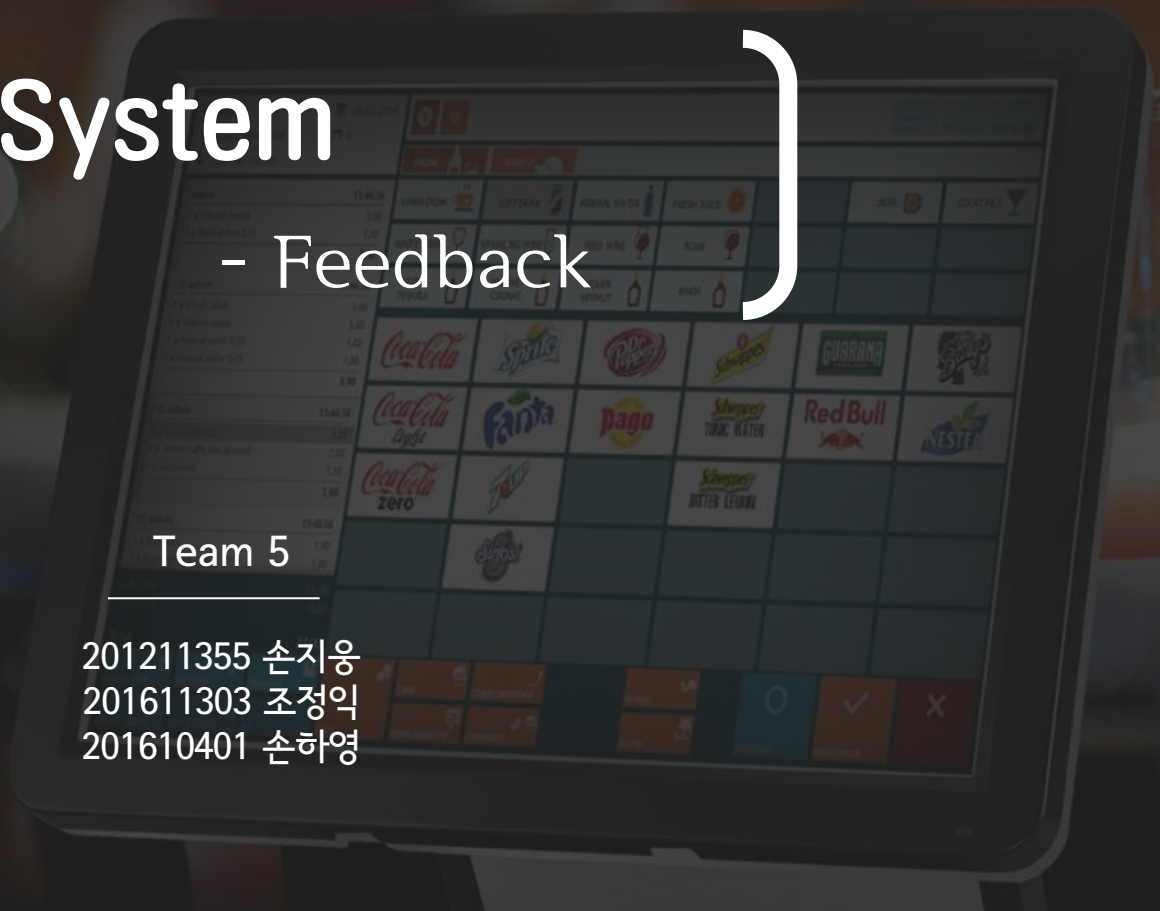
POS System

- Feedback



Team 5

201211355 손지웅
201611303 조정익
201610401 손하영

CONTENTS,

1. Fail Case
2. Answer
3. Re-test result
4. 소감

1. Fail Case

1.1 Case #1

1.2 Case #2

1.3 Case #3

1.4 Case #4

1.1 Case #1

T4_POS.STP.000.001	바코드로 과자(001)추가 후 목록제거를 통해 0으로변경, 라면과 물을 수량1로 추가한다. 돈을 지불하고 판매 완료 뒤, 환불실시.	판매영수증에 과자가 없고 라면과 물만 있는지 확인. 영수증에도 과자가 없고 라면과 물만 1개씩인지 확인. 환불 후 환불되었는지 확인. 재고서버에 과자,물 라면 모두 수량이 100개인지 확인	Fail

1.2 Case #2

3_000_000	과자(001)의 재고를 98개로 맞춰놓고 하루(3분)이 넘어갈 때를 기다린다.	상품 파일, 판매 관리 파일이 다음 날짜로 새로 생성된다.
3_000_001	1_000_006의 상태에서 판매를 하고 하루(3분)이 넘어갈 때를 기다린다.	상품 파일, 판매 관리 파일이 다음 날짜로 새로 생성된다.

3_000_000	Fail (판매 목록 파일이 날짜가 지나도 새로 생성안됨)
3_000_001	Fail('')

1.3 Case #3

T4_POS.STP.003.001	메인화면에서 환불 입력한다. 그 다음, 환불되지 않은 영수증 입력후 환불확정을 입력한다.	환불기능이 제대로 작동하는지 확인	Fail

2_000_000	Fail (환불 Y/N 선택 불가능)

1.4 Case #4

T4_POS.STP.000.001	바코드로 과자(001)추가 후 목록제거를 통해 0으로변경, 라면과 물을 수량1로 추가한다. 돈을 지불하고 판매 완료 뒤, 환불실시.	판매영수증에 과자가 없고 라면과 물만 있는지 확인. 영수증에도 과자가 없고 라면과 물만 1개씩인지 확인. 환불 후 환불되었는지 확인. 재고서버에 과자,물 라면 모두 수량이 100개인지 확인	Fail

2. Answer

2.1 Case #1 (목록 제거)

2.2 Case #2 (판매 관리 목록)

2.3 Case #3 (환불 유무)

2.4 Case #4 (판매 시 재고 부족)

2.1 Case #1 (목록 제거)

```c
while (1) {
    printf("1. (+) 2. (-) : 3. Delete : ");
    scanf("%d", &sel);

    if(sel < 1 || sel > 3) {
        printf("Wrong Input!\n\n");
        continue;
    }
    else {
        break;
    }
}
```

```c
else {
    if (sale_info->s_arr[index - 1].quantity == 0) {
        // message passing 부분
        printf("Wrong Input!\n\n");
        return;
    }
    int p_num = 0;
    p_num = sale_info->s_arr[index - 1].quantity;
    sale_info->s_arr[index - 1].quantity = 0;
    sale_info->total_price -= (sale_info->s_arr[index - 1].price * p_num);
    stock_info->s_arr[index - 1].quantity += p_num;
    sale_display(sale_info);
}
```

2.1 Case #1 (목록 제거)

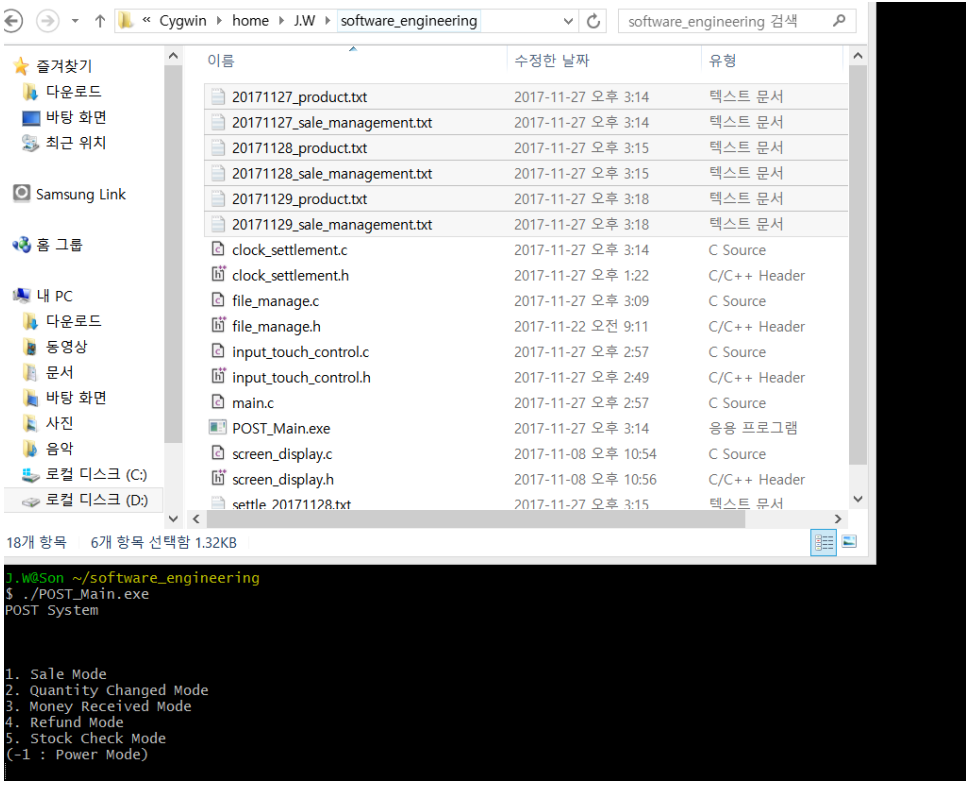
```
Casher Screen
     snack      quantity :       4 | price :    1000 | sum_price :    4000
 ice_cream      quantity :       1 | price :    1500 | sum_price :    1500
     water      quantity :       1 | price :     500 | sum_price :     500
     ramen      quantity :       1 | price :     800 | sum_price :     800
  beverage      quantity :       2 | price :    1200 | sum_price :    2400
    coffee      quantity :       2 | price :    2000 | sum_price :    4000
Total Sale Money                                                      13200
Receive Money                                                             0
Send Money                                                                0


Client Screen
Total Sale Money                                                      13200
Receive Money                                                             0
Send Money                                                                0


1. Sale Mode
2. Quantity Changed Mode
3. Money Received Mode
4. Refund Mode
5. Stock Check Mode
(-1 : Power Mode)
2

Quantity Change Mode
1. snack 2.ice cream 3.fruit 4.water 5.ramen 6.beverage 7.coffee
-> 6
1. (+) 2. (-) : 3. Delete : 3


Casher Screen
     snack      quantity :       4 | price :    1000 | sum_price :    4000
 ice_cream      quantity :       1 | price :    1500 | sum_price :    1500
     water      quantity :       1 | price :     500 | sum_price :     500
     ramen      quantity :       1 | price :     800 | sum_price :     800
    coffee      quantity :       2 | price :    2000 | sum_price :    4000
Total Sale Money                                                      10800
Receive Money                                                             0
Send Money                                                                0


Client Screen
Total Sale Money                                                      10800
Receive Money                                                             0
Send Money                                                                0
```

2.1 Case #1 (목록 제거)

```
Money Received Mode

Received Money : 11000



Casher Screen
     snack      quantity :       4 | price :    1000 | sum_price :    4000
 ice_cream      quantity :       1 | price :    1500 | sum_price :    1500
     water      quantity :       1 | price :     500 | sum_price :     500
     ramen      quantity :       1 | price :     800 | sum_price :     800
    coffee      quantity :       2 | price :    2000 | sum_price :    4000
Total Sale Money                                                      10800
Receive Money                                                         11000
Send Money                                                              200


Client Screen
Total Sale Money                                                      10800
Receive Money                                                         11000
Send Money                                                              200


1. Sale Mode
2. Quantity Changed Mode
3. Money Received Mode
4. Refund Mode
5. Stock Check Mode
(-1 : Power Mode)
5
Stock Check Mode


Casher Screen
     snack      quantity :      96 | price :    1000
 ice_cream      quantity :      99 | price :    1500
     fruit      quantity :     100 | price :    3000
     water      quantity :      99 | price :     500
     ramen      quantity :      99 | price :     800
  beverage      quantity :     100 | price :    1200
    coffee      quantity :      98 | price :    2000
```

2.2 Case #2 (판매 관리 목록)

```
init_arr(s_r);
init_stock_info(&stock_info);
make_stock_file(&stock_info, year, month, day);
sale_list_file(&sale_info, year, month, day, hour, min);
```

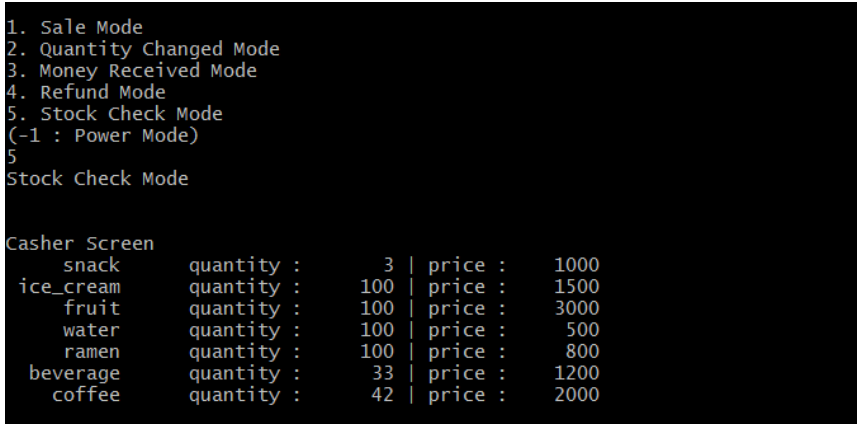


2.3 Case #3 (환불 유무)

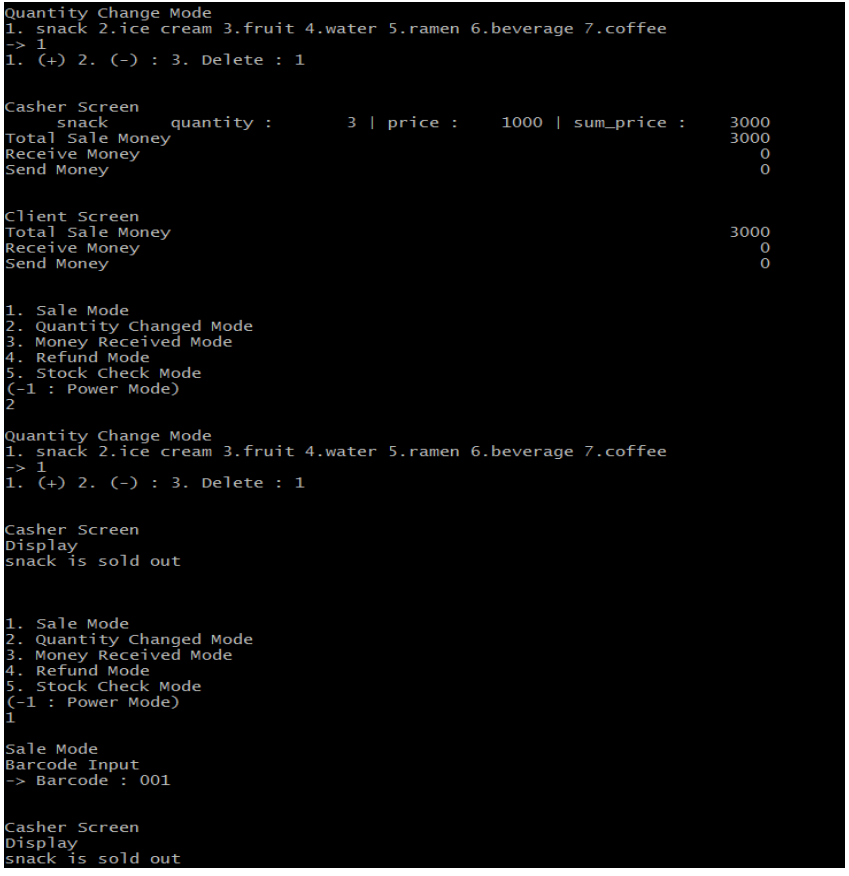
```c
while (1) {
    printf("1. Refund Ok 2. Refund Cancel : ");
    scanf("%d", &mode);
    if (mode == 2) {
        return 0;
    }
    else if (mode == 1) {
        break;
    }
    else {
        printf("Wrong Input!\n\n");
        continue;
    }
}
```

```
1. Sale Mode
2. Quantity Changed Mode
3. Money Received Mode
4. Refund Mode
5. Stock Check Mode
(-1 : Power Mode)
4
Refund Mode
Refund Receipt Barcode : 201711302331
1. Refund Ok 2. Refund Cancel :
```

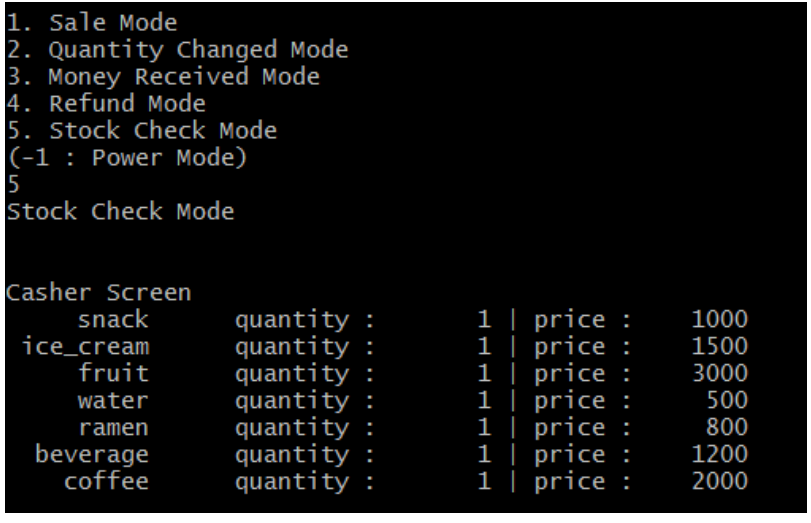
2.4 Case #4 (판매 시 재고 부족)



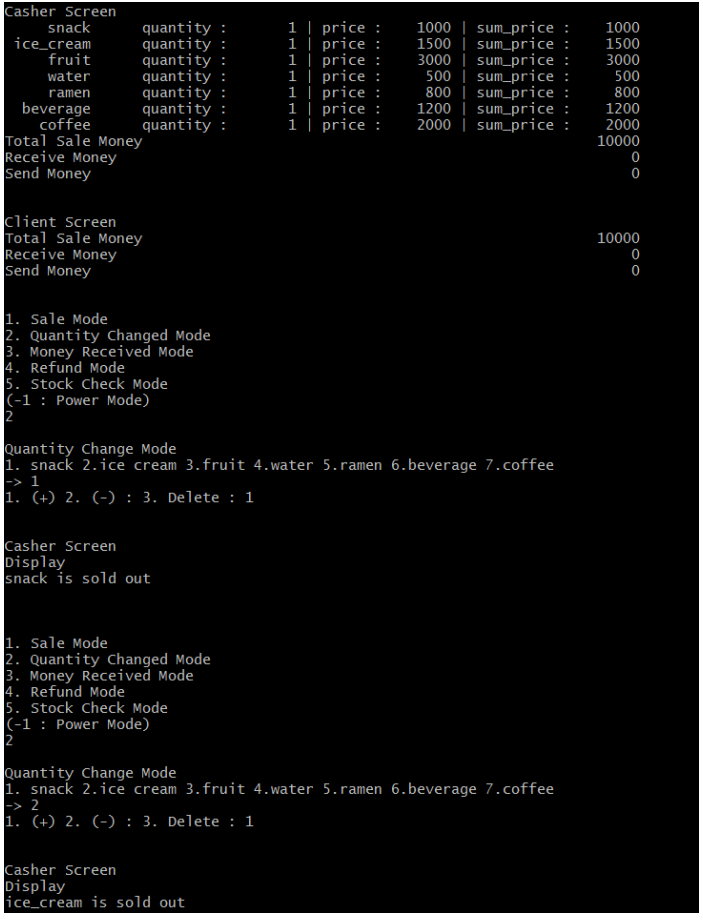
2.4 Case #4 (판매 시 재고 부족)



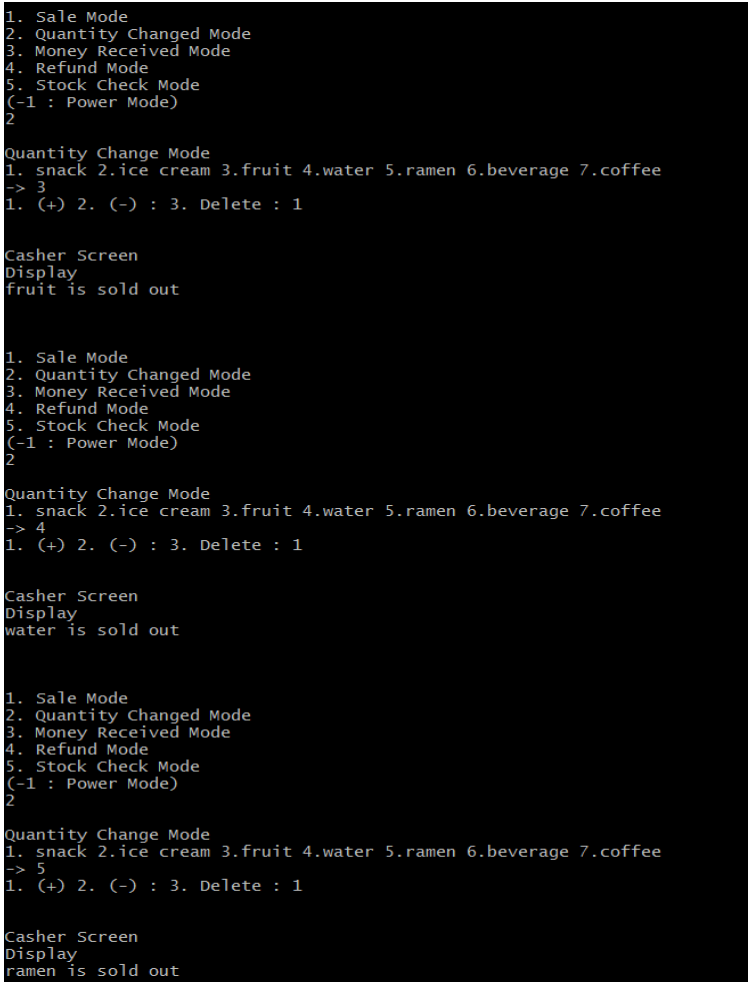
2.4 Case #4 (판매 시 재고 부족)



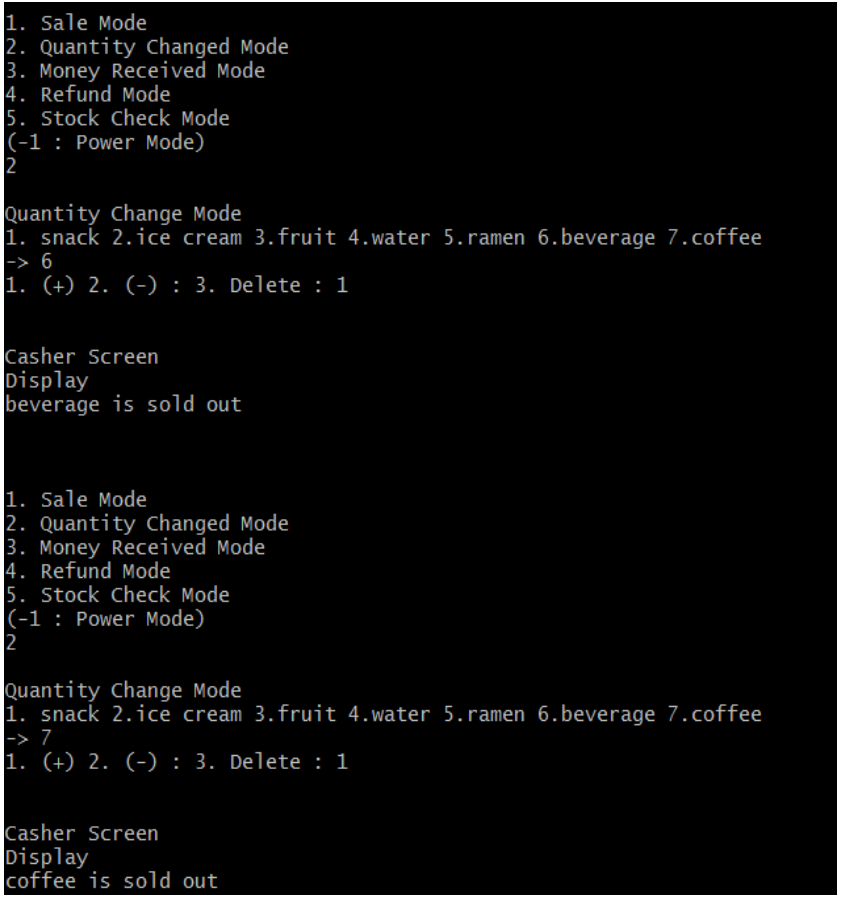
2.4 Case #4 (판매 시 재고 부족)



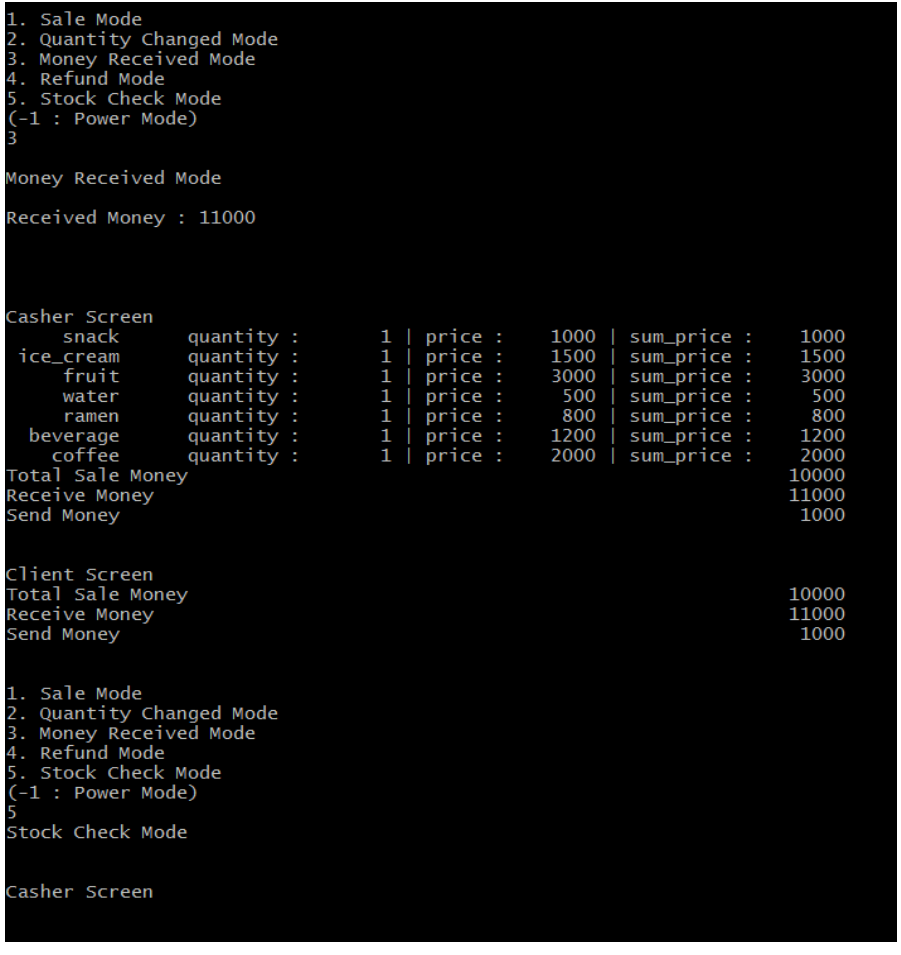
2.4 Case #4 (판매 시 재고 부족)



2.4 Case #4 (판매 시 재고 부족)



2.4 Case #4 (판매 시 재고 부족)



3. RE-test result

3.1 result (0)

3.2 result (1)

3.3 result (2)

3.4 result (3)

3.5 result (4)

코드를 바꾸었기 때문에 다른 곳에서 오류가 발생할 수 있음

⇒ System test 를 다시 진행

3.1 result (0_000)

Id	Input	Output (Expected)
0_000_000	판매 모드 진행 중 전원을 종료한다.	전원이 종료된다.
0_000_001	환불 모드 진행 중 전원을 종료한다.	전원이 종료된다.
0_000_002	재고 확인 모드 진행 중 전원을 종료한다.	전원이 종료된다.

Id	Result
0_000_000	Passed
0_000_001	Passed
0_000_002	Passed

3.2 result(1_000)

Id	Input	Output (Expected)
1_000_000	바코드 001 입력	과자(001)가 추가
1_000_001	바코드 010 입력	아이스크림(010)가 추가
1_000_002	바코드 011 입력	과일(011)이 추가
1_000_003	바코드 100 입력	물(100)이 추가
1_000_004	바코드 101 입력	라면(101)이 추가
1_000_005	바코드 110 입력	음료수(110)이 추가
1_000_006	바코드 111 입력	커피(111)가 추가
1_000_007	바코드 000 입력	예외 처리
1_000_008	바코드 001 입력 후 001 입력	과자(001)가 2개 추가된다.

Id	Result
1_000_000	Passed
1_000_001	Passed
1_000_002	Passed
1_000_003	Passed
1_000_004	Passed
1_000_005	Passed
1_000_006	Passed
1_000_007	Passed
1_000_008	Passed

3.2 result(1_001 ~ 1_002)

Id	Input	Output (Expected)
1_001_000	1_000_000상태에서 과자를 3개 추가	과자(001)가 5개가 된다
1_001_001	1_001_000상태에서 과자를 4개 감소	과자(001)가 1개가 된다
1_001_002	1_001_001상태에서 과자를 1개 감소시켜 제거	과자(001)이 목록에서 제거된다
1_001_003	1_001_002상태에서 과자를 1개 감소	예외 처리
1_002_000	하루가 지나기 전에 001 상품을 102개 판매하려고 한다.	"개수가 부족합니다"
1_003_000	1_000_000상태에서 판매 목록의 합계 < 고객에게 받은 돈	결제가 진행된다.
1_003_001	1_000_000상태에서 판매 목록의 합계 > 고객에게 받은 돈	결제가 진행되지 않는다.

Id	Result
1_001_000	Passed
1_001_001	Passed
1_001_002	Passed
1_001_003	Passed

Id	Result
1_002_000	Passed
1_003_000	Passed
1_003_001	Passed

3.2 result(1_004 ~ 1_010)

Id	Input	Output (Expected)
1_004_000	1_003_000	캐셔 화면과 고객 화면에 판매 정보가 업데이트된다.
1_005_000	1_003_000	sale_YYYYMMDDhhmm.txt 파일이 생성된다.
1_006_000	1_003_000	YYYYMMDD_product.txt의 상품의 개수가 업데이트된다.
1_007_000	1_003_000	YYYYMMDD_sale_management.txt에 현재 판매 정보가 추가된다.
1_008_000	1_003_000	캐셔 화면(터치 스크린) Display example 총 판매액 : 20,000 현금 : 30,000 거스름돈 : 10,000 상품 : 수량 : 5 / 단가 : 500 / 판매 금액 : 2,500 ... 상품 : 수량 : 4 / 단가 : 1,000 / 판매 금액 : 4,000
1_009_000	1_003_000	고객 화면 Display example 총 판매액 : 20,000 현금 : 30,000 거스름돈 : 10,000
1_010_000	1_003_000	영수증 양식 영수증 번호 : YYYY.MM.DD.hh.mm (년.월.일.시.분) 날짜 : YYYY.MM.DD (년.월.일) 판매 상품 상품, 단가, 수량, 판매 금액 ... 상품, 단가, 수량, 판매 금액 총 판매액 : x

3.2 result(1_004 ~ 1_010)

Id	Result
1_004_000	Passed
1_005_000	Passed
1_006_000	Passed
1_007_000	Passed
1_008_000	Passed
1_009_000	Passed
1_010_000	Passed

3.3 result (2_000 ~ 2_001)

Id	Input	Output (Expected)
2_000_000	1_003_000에서 발급했던 영수증 바코드를 입력하고 환불 취소를 입력한다.	환불이 되지 않는다.
2_000_001	2_000_000에 이어 1_003_000에서 발급했던 영수증 바코드를 입력하고 환불을 완료한다.	환불이 된다.
2_000_002	2_000_001에 이어 1_003_000에서 발급했던 영수증 바코드를 입력한다.	"환불 불가능"
2_000_003	1_000_006에 이어 판매를 종료한 후 발급했던 영수증 바코드를 입력하고 환불을 완료한다.	환불이 된다.
2_000_004	2_000_003에 이어 1_000_006에서 발급했던 영수증 바코드를 입력한다.	"환불 불가능"
2_001_000	발급하지 않았던 임의의 영수증 바코드를 입력한다.	"환불 불가능" 메시지가 나온다.

Id	Result
2_000_000	Passed
2_000_001	Passed
2_000_002	Passed
2_000_003	Passed
2_000_004	Passed
2_001_000	Passed

3.3 result (2_002 ~ 2_006)

Id	Input	Output (Expected)
2_002_000	2_000_001	refund_YYYYMMDDhhmm.txt 을 생성
2_002_001	2_000_003	refund_YYYYMMDDhhmm.txt 을 생성
2_003_000	2_000_001	YYYYMMDD_product.txt의 상품의 개수를 업데이트한다.
2_003_001	2_000_003	YYYYMMDD_product.txt의 상품의 개수를 업데이트한다.
2_004_000	2_000_001	캐셔 화면(터치 스크린) Display example 판매 날짜 : 2017년 09월 07일 05시 20분 환불 금액 : 20,000 상품 : 수량 : 5 / 단가 : 500 / 판매 금액 : 2,500 ... 상품 : 수량 : 4 / 단가 : 1,000 / 판매 금액 : 4,000
2_005_000	2_000_001	고객 화면 Display example 판매 날짜 : 2017년 09월 07일 05시 20분 환불 금액 : 20,000
2_006_000	2_000_001	영수증 양식 영수증 번호 : YYYY.MM.DD.hh.mm (년.월.일.시.분) 날짜 : YYYY.MM.DD (년.월.일) 환불 상품 상품, 단가, 수량, 판매 금액 ... 상품, 단가, 수량, 판매 금액 환불 금액 : x

3.3 result (2_002 ~ 2_006)

Id	Result
2_002_000	Passed
2_002_001	Passed
2_003_000	Passed
2_003_001	Passed
2_004_000	Passed
2_005_000	Passed
2_006_000	Passed

3.4 result (3_000 ~ 3_004)

Id	Input	Output (Expected)
3_000_000	과자(001)의 재고를 98개로 맞춰놓고 하루(3분)이 넘어갈 때를 기다린다.	상품 파일, 판매 관리 파일이 다음 날짜로 새로 생성된다.
3_000_001	1_000_006의 상태에서 판매를 하고 하루(3분)이 넘어갈 때를 기다린다.	상품 파일, 판매 관리 파일이 다음 날짜로 새로 생성된다.
3_001_000	과자(001)의 재고를 98개로 맞춰놓고 하루(3분)이 넘어갈 때를 기다린다.	상품파일의 모든 상품은 100개로 초기화한다.
3_001_001	1_000_006의 상태에서 판매를 하고 하루(3분)이 넘어갈 때를 기다린다.	상품파일의 모든 상품은 100개로 초기화한다.
3_002_000	하루(3분)이 지나기 전에 과자(001)의 재고를 98개로 맞춰놓고 전원을 껐다 키고 재고 확인을 한다.	캐셔 화면에 재고 정보가 출력되고 stock_YYYYMMDD.txt파일을 이용해 재고 정보를 출력한다. 당일 재고 정보이므로 과자의 재고는 98개여야 한다.
3_002_001	하루(3분)이 지나기 전에 1_000_006의 상태에서 판매를 하고 전원을 껐다 키고 재고 확인을 한다.	캐셔 화면에 재고 정보가 출력되고 stock_YYYYMMDD.txt파일을 이용해 재고 정보를 출력한다. 당일 재고 정보이므로 상품의 재고는 99개여야 한다.
3_003_000	판매 중일 때 재고 확인이 되는지 확인한다.	불가능해야 한다.
3_003_001	환불 중일 때 재고 확인이 되는지 확인한다.	불가능해야 한다.
3_004_000	3_001_000	재고 확인 양식 날짜 : YYYY.MM.DD.hh.mm (년.월.일.시.분) 상품, 단가, 재고 ... 상품, 단가, 재고

3.4 result (3_000 ~ 3_004)

Id	Result
3_000_000	Passed
3_000_001	Passed
3_001_000	Passed
3_001_001	Passed
3_002_000	Passed
3_002_001	Passed
3_003_000	Passed
3_003_001	Passed
3_004_000	Passed

3.5 result (4_000 ~ 4_004)

Id	Input	Output (Expected)
4_000_000	프로그램 종료하기 전에 시간을 체크하고 재실행 시 기존 시각이 이어지는지 체크한다.	기존 시각이 이어져야 한다.
4_001_000	현실 시간 3분이 지나갔을 때 하루가 넘어가며 settle_YYYYMMDD.txt가 생성되는지 확인한다. (기본화면)	하루가 넘어가야 하며 settle_YYYYMMDD.txt가 생성되는지 확인한다.
4_001_001	현실 시간 6분이 지나가고 settle_YYYYMMDD.txt가 생성되는지 확인한다. (기본화면)	settle_YYYYMMDD.txt가 생성되는지 확인한다.
4_001_002	현실 시간 9분이 지나가고 settle_YYYYMMDD.txt가 생성되는지 확인한다. (기본화면)	settle_YYYYMMDD.txt가 생성되는지 확인한다.
4_002_000	하루가 지나가기 전에 1_003_000을 진행하고 4_001_000을 진행한다.	settle_YYYYMMDD.txt가 생성되며 1_003_000의 정보가 있어야 한다.
4_003_000	프로그램을 판매 화면으로 두고 3분을 기다린다.	정산이 진행되지 않는다.
4_003_001	4_003_000의 상태에서 판매 모드를 완료한다	정산이 진행된다.
4_003_002	프로그램을 환불 화면으로 두고 3분을 기다린다.	정산이 진행되지 않는다.
4_003_003	4_003_001의 상태에서 환불 모드를 완료한다	정산이 진행된다.
4_004_000	4_002_000	
4_004_001	4_003_003	"""

3.5 result (4_000 ~ 4_004)

Id	Result
4_000_000	Passed
4_001_000	Passed
4_001_001	Passed
4_001_002	Passed
4_002_000	Passed
4_003_000	Passed
4_003_001	Passed
4_003_002	Passed
4_003_003	Passed
4_004_000	Passed
4_004_001	Passed

4. 소감

소프트웨어 공학이라는 과목은 얼핏 보면 애매해 보이지만 결국 사람에 관한 학문이라고 느꼈습니다. 팀원들과 효율적으로 소통하고 화합하는 것이 소프트웨어 공학에서 가장 중요하다고 생각합니다.

201611303 조정익

소프트웨어 공학개론 실습을 진행하며 처음엔 너무 시간이 빠듯함을 느꼈습니다, 하지만 코드가 완성 된 이후는 다소 여유가 생겨서 소공 실습 스케쥴을 조금 조정한다면 잘 분배되어 넉넉하게 진행할 수 있을 거라 생각되고 보람찬 수업이었습니다. 그리고 다시금 팀플의 어려움을 느끼게 되는 계기가 되었습니다.

201211355 손지웅

소프트웨어 공학 수업을 들으면서 효율적인 소프트웨어 개발 방법뿐만 아니라 그 외의 여러 가지를 배울 수 있었습니다. 다음부터는 코드~ 유닛 테스트 기간을 좀 더 길게 잡아주셨으면 좋겠습니다. 최선을 다할 수 있어 보람찬 수업이였습니다.

201610401 손하영

(Q & A)”.