JooHyun – Lee (comkiwer)

TS마트관리

Hancom Education Co. Ltd.

Problem

Problem

마트 관리 시스템이 있다. 마트는 상품을 구매해서 판매한다.

상품을 구매할 때는 구매 ID와 함께 상품 번호, 구매 가격, 수량이 주어진다.
상품 번호는 상품을 구별하는 값으로 동일한 상품 번호로 여러 번 구매가 가능하다.

구매를 취소할 때는, 구매 ID만 주어진다.
구매 ID에 해당하는 상품 수량이 모두 남아 있을 경우에만, 취소가 가능하다.

Problem

상품을 판매할 때는 판매 ID와 함께 상품 번호, 판매 가격, 수량이 주어진다.
가장 싸게 구매한 상품부터 판매한다.
가격이 동일한 경우에는 구매 ID 값이 작은 상품부터 판매한다.

판매한 상품을 환불해 줄 때는, 판매 ID만 주어진다.
판매 ID로 판매되었던 모든 상품이 다시 재고가 된다.

위와 같이 동작하는 마트 관리 시스템을 구현하여 보자.

아래 API 설명을 참조하여 각 함수를 구현하라.

Problem

void init() :

각 테스트 케이스의 처음에 호출된다.

재고가 없는 상태가 된다.

int buy(int BID, int product, int price, int quantity) :

product 상품을 price 가격으로 quantity 개 구매한다. 구매 ID는 BID이다.

구매 후에, 마트가 보유 중인 product 상품의 총 개수를 반환한다.

Parameters

BID: 구매 ID (1 ≤ BID ≤ 1,000,000,000)

product: 상품 번호 (1 ≤ product ≤ 1,000,000,000)

price: 구매 가격 (1,000 ≤ price ≤ 300,000)

quantity: 구매 수량 (10 ≤ quantity ≤ 100)

Returns

product 상품의 재고 수량을 반환한다.

Problem

int cancel(int BID) :

구매 ID가 BID인 구매를 취소한다.

BID로 구매했던 상품 수량이 모두 마트에 남아 있을 경우에만, 취소가 가능하다.

한 개라도 부족하다면 취소에 실패하고 -1을 반환한다.

BID로 구매한 상품이 판매 된 적이 있더라도,
환불을 통해 다시 모두 재고가 되었다면 취소가 가능함을 유의하라.

BID로 구매한 내역이 없거나, 이미 취소한 구매 ID라면, 취소에 실패하고 -1을 반환한다.

취소가 가능하다면, 재고에서 BID로 구매했던 상품을 삭제하고,
마트에 남아 있는 동일 상품의 개수를 반환한다.

Parameters

BID: 구매 ID (1 ≤ BID ≤ 1,000,000,000)

Returns

취소에 실패할 경우, -1을 반환한다.

취소에 성공할 경우, 취소된 상품과 동일한 상품이 마트에 얼마나 남아있는지 그 개수를 반환한다.

Problem

int sell(int SID, int product, int price, int quantity) :

product 상품을 price 가격으로 quantity 개 판매한다. 판매 ID는 SID이다.

판매 ID와 구매 ID는 서로 독립적인 ID이기 때문에,
판매 ID 값과 구매 ID 값이 서로 동일한 경우도 있다.

product 상품의 재고 수량이 quantity 보다 작다면, 판매에 실패하고 -1을 반환한다.

판매가 가능하다면, 가장 싸게 구매한 상품부터 판매한다.

가격이 동일한 경우에는 구매 ID 값이 작은 상품부터 판매한다.

판매 후에, 총 판매 수익을 반환한다. 개당 판매 수익은 판매 가격에서 구매 가격을 뺀 값이다.

Parameters

SID: 판매 ID (1 ≤ SID ≤ 1,000,000,000)

product: 상품 번호 (1 ≤ product ≤ 1,000,000,000)

price: 판매 가격 (2,000 ≤ price ≤ 300,000)

quantity: 판매 수량 (1 ≤ quantity ≤ 500)

Returns

판매에 실패할 경우, -1을 반환한다.

판매에 성공할 경우, SID 판매로 발생한 총 수익을 반환한다.

Problem

int refund(int SID) :

SID로 판매한 상품에 대해 환불해 준다.

환불해 준 상품의 총 개수를 반환한다.

환불해 준 상품은 모두 재고로 쌓인다.

SID로 판매한 내역이 없거나, 이미 환불해준 판매 ID라면, 환불에 실패하고 -1을 반환한다.

Parameters

SID: 판매 ID (1 ≤ SID ≤ 1,000,000,000)

Returns

환불에 실패할 경우, -1을 반환한다.

환불에 성공할 경우, 환불해 준 상품의 총 개수를 반환한다.

Problem

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.
2. 각 테스트 케이스에서 상품 종류는 600 이하이다.
3. 각 테스트 케이스에서 buy() 함수의 호출 횟수는 30,000 이하이다.
4. 각 테스트 케이스에서 sell() 함수의 호출 횟수는 30,000 이하이다.
5. 각 테스트 케이스에서 모든 함수의 호출 횟수 총합은 80,000 이하이다.

Problem analysis

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

(순서 1) 초기에는 재고가 없다.

(순서 2) 100 상품을 개당 가격 2,000으로 30개 구매한다.
구매 ID는 100이다.

100 상품의 재고 수량으로 30을 반환한다.

(순서 3) 100 상품을 개당 가격 2,500으로 50개 구매한다.
구매 ID는 500이다.

이전 재고 수량 30에 50개가 추가되므로,
100 상품의 재고 수량으로 80을 반환한다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

(순서 4) 300 상품을 개당 가격 1,900으로 60개 구매한다.
구매 ID는 200이다.

300 상품의 재고 수량으로 60을 반환한다.

(순서 5) 100 상품을 개당 가격 2,000으로 40개 구매한다.
구매 ID는 400이다.

이전 재고 수량 80에 40개가 추가되므로,
100 상품의 재고 수량으로 120을 반환한다.

(순서 6) 100 상품을 개당 가격 1,800으로 20개 구매한다.
구매 ID는 700이다.

이전 재고 수량 120에 20개가 추가되므로,
100 상품의 재고 수량으로 140을 반환한다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 1]

Product	stock
100	price=1800(BID=700, quantity=20) price=1900(BID=600, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=60)

(순서 7) 100 상품을 개당 가격 1,900으로 10개 구매한다.
구매 ID는 600이다.

이전 재고 수량 140에 10개가 추가되므로,
100 상품의 재고 수량으로 150을 반환한다.

[Fig. 1]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 2]

Product	stock
100	price=1800(BID=700, quantity=10) price=1900(BID=600, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=60)

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)

((순서 8) 100 상품을 개당 가격 3,000으로 10개 판매한다. 판매 ID는 200이다. 가장 싸게 구매한 100 상품은 700 구매 ID로 구매한 상품으로 20개가 있다. 판매 가격(3,000)에서 구매 가격(1,800)을 빼면 개당 판매 수익은 1,200이다.

10개를 판매하기 때문에, 총 판매 수익으로 12,000을 반환한다.

[Fig. 2]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 2]

Product	stock
100	price=1800(BID=700, quantity=10) price=1900(BID=600, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=60)

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)

(순서 9) 구매 ID가 700인 구매를 취소하려고 한다.

700 구매 ID로 주문했던 상품은 모두 20개인데,

재고로 10개만 남아있기 때문에, 취소에 실패하고 -1을 반환한다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 3]

Product	stock
100	price=1800(BID=700, quantity=10) ~~price=1900(BID=600, quantity=10)~~ price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=60)

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)

(순서 10) 구매 ID가 600인 구매를 취소하려고 한다.

600 구매 ID로 주문했던 상품은 모두 10개인데,

재고로 모두 남아있기 때문에, 취소에 성공한다.

취소하고 남은 동일 상품의 개수로 130을 반환한다.

[Fig. 3]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 3]

Product	stock
100	price=1800(BID=700, quantity=10) ~~price=1900(BID=600, quantity=10)~~ price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=60)

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)

(순서 11) 300 상품을 개당 가격 2,000으로 100개 판매하려고 한다. 판매 ID는 500이다.

300 상품은 60개밖에 없기 때문에, 판매에 실패하고 −1을 반환한다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 4]

Product	stock
100	price=1800(BID=700, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	**price=1900(BID=200, quantity=30)**

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)
400	**product=300(price=1900, BID=200, quantity=30)**

(순서 12) 300 상품을 개당 가격 2,000으로 30개 판매한다. 판매 ID는 400이다.

판매 가격(2,000)에서 구매 가격(1,900)을 빼면 개당 판매 수익은 100이다.

30개를 판매하기 때문에, 총 판매 수익으로 3,000을 반환한다.

[Fig. 4]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 5]

Product	stock
100	price=2000(BID=400, quantity=30) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
200	product=100(price=1800, BID=700, quantity=10)
400	product=300(price=1900, BID=200, quantity=30)
100	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=100, quantity=30) product=100(price=2000, BID=400, quantity=10)

(순서 13) 100 상품을 개당 가격 3,500으로 50개 판매한다.
판매 ID는 100이다. 가장 싸게 구매한 순서대로 팔면, 1,800 가격에 구매한 상품 10개(구매 ID=700), 2,000 가격에 구매한 상품 30개(구매 ID=100), 2,000 가격에 구매한 상품 10개(구매 ID=400)를 팔게 된다.
판매 수익은 각각 (3,500 – 1,800) x 10 = 17,000,
(3,500 – 2000) x 30 = 45,000,
(3,500 – 2000) x 10 = 15,000이다.
총 판매 수익으로 77,000을 반환한다.
[Fig. 5]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 6]

Product	stock
100	price=1800(BID=700, quantity=10) price=2000(BID=400, quantity=30) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
~~200~~	~~product=100(price=1800, BID=700, quantity=10)~~
400	product=300(price=1900, BID=200, quantity=30)
100	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=100, quantity=30) product=100(price=2000, BID=400, quantity=10)

(순서 14) 200 판매 ID로 판매된 상품을 환불해 준다.

200 판매 ID로 판매된 상품 번호는 100이고 10개가 판매됐다.

10개가 다시 재고가 되고,
환불해 준 상품의 개수로 10을 반환한다.

[Fig. 6]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 7]

Product	stock
100	price=2000(BID=400, quantity=20) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
400	product=300(price=1900, BID=200, quantity=30)
100	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=100, quantity=30) product=100(price=2000, BID=400, quantity=10)
300	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=400, quantity=10)

(순서 15) 100 상품을 개당 가격 2,500으로 20개 판매한다. 판매 ID는 300이다.

가장 싸게 구매한 순서대로 팔면, 1,800 가격에 구매한 상품 10개(구매 ID=700),

2,000 가격에 구매한 상품 10개(구매 ID=400)를 팔게 된다.

판매 수익은 각각 (2,500 − 1,800) x 10 = 7,000, (2,500 − 2000) x 10 = 5,000이다.

총 판매 수익으로 12,000을 반환한다.

[Fig. 7]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 8]

Product	stock
100	price=1800(BID=700, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=30) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
400	product=300(price=1900, BID=200, quantity=30)
100	~~product=100(price=1800, BID=700, quantity=10)~~ ~~product=100(price=2000, BID=100, quantity=30)~~ ~~product=100(price=2000, BID=400, quantity=10)~~
300	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=400, quantity=10)

(순서 16) 100 판매 ID로 판매된 상품을 환불해 준다.

100 판매 ID로 판매된 상품 번호는 100이고 50개가 판매됐다.

50개가 다시 재고가 되고,
환불해 준 상품의 개수로 50을 반환한다.

[Fig. 8]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 8]

Product	stock
100	price=1800(BID=700, quantity=10) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=30) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
400	product=300(price=1900, BID=200, quantity=30)
100	~~product=100(price=1800, BID=700, quantity=10)~~ ~~product=100(price=2000, BID=100, quantity=30)~~ ~~product=100(price=2000, BID=400, quantity=10)~~
300	product=100(price=1800, BID=700, quantity=10) product=100(price=2000, BID=400, quantity=10)

(순서 17) 구매 ID가 400인 구매를 취소하려고 한다.
400 구매 ID로 주문했던 상품은 모두 40개인데,
재고로 30개만 남아있기 때문에,
취소에 실패하고 -1을 반환한다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 9]

Product	stock
100	price=1800(BID=700, quantity=20) price=2000(BID=100, quantity=30) price=2000(BID=400, quantity=40) price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
400	product=300(price=1900, BID=200, quantity=30)
~~300~~	~~product=100(price=1800, BID=700, quantity=10)~~ ~~product=100(price=2000, BID=400, quantity=10)~~

(순서 18) 300 판매 ID로 판매된 상품을 환불해 준다.

300 판매 ID로 판매된 상품 번호는 100이고 20개가 판매됐다.

20개가 다시 재고가 되고,
환불해 준 상품의 개수로 20을 반환한다.

[Fig. 9]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 예제

[Table 1]

순서	Function	return	Figure
1	init()		
2	buy(100, 100, 2000, 30)	30	
3	buy(500, 100, 2500, 50)	80	
4	buy(200, 300, 1900, 60)	60	
5	buy(400, 100, 2000, 40)	120	
6	buy(700, 100, 1800, 20)	140	
7	buy(600, 100, 1900, 10)	150	Fig. 1
8	sell(200, 100, 3000, 10)	12000	Fig. 2
9	cancel(700)	-1	
10	cancel(600)	130	Fig. 3
11	sell(500, 300, 2000, 100)	-1	
12	sell(400, 300, 2000, 30)	3000	Fig. 4
13	sell(100, 100, 3500, 50)	77000	Fig. 5
14	refund(200)	10	Fig. 6
15	sell(300, 100, 2500, 20)	12000	Fig. 7
16	refund(100)	50	Fig. 8
17	cancel(400)	-1	
18	refund(300)	20	Fig. 9
19	cancel(400)	100	Fig. 10

[Fig. 10]

Product	stock
100	price=1800(BID=700, quantity=20) price=2000(BID=100, quantity=30) ~~price=2000(BID=400, quantity=40)~~ price=2500(BID=500, quantity=50)
300	price=1900(BID=200, quantity=30)

SID	product sold
400	product=300(price=1900, BID=200, quantity=30)

(순서 19) 구매 ID가 400인 구매를 취소하려고 한다.
400 구매 ID로 주문했던 상품은 모두 40개인데,
재고로 모두 남아있기 때문에, 취소에 성공한다.
취소하고 남은 동일 상품의 개수로 100을 반환한다.

[Fig. 10]은 함수 호출의 결과를 나타낸 그림이다.

Problem analysis : 제약조건 및 API

- 구매 아이디(BID), 상품 번호(product), 판매 아이디(SID)의 범위가
  1 ~ 10억 이므로 renumbering이 필요할 수 있다.

- 상품 별로 판매가 이루어지며
  우선순위는 다음과 같다.
  1. 가격의 오름차순
  2. 구매 아이디(BID)의 오름차순

- 상품 종류는 600 이하이다.
- buy(), sell() 함수의 호출은 각각 30,000 이하이다.
- 모든 함수 호출은 80,000이하이다.

Problem analysis : 제약조건 및 API

- 구매는 구매 아이디로 판매는 상품 번호를 기준으로 처리된다.
  또한 판매는 우선순위에 따라 처리되어야 한다.
  이를 어떻게 처리할 것인가?

- 구매에 대한 취소처리가 필요하다.
- 판매에 대한 취소처리가 필요하다.

- 구매, 판매, 구매 취소, 판매 취소에 따라 구매 아이디 별, 상품 번호 별
  재고의 개수가 달라진다.
- 이들을 어떻게 처리할 것인가?

Solution sketch

Solution sketch

- 구매 아이디(BID)와 상품 번호(product) 는 renumbering 하여 사용하는 것으로 한다.

```
// 구입번호, 제품번호 renumbering id를 위한 hash table
unordered_map<int, int> bHash, pHash;
```

- renumbering 한 구매 아이디를 bid,
  renumbering한 상품 번호 id를 pid라 할 수 있다.

- pid별 상품 개수를 prodSum[605]에 담아 관리 할 수 있다.

- 판매 아이디별 판매 정보를 다음과 같은 자료구조로 관리 할 수 있다.

```
// SID를 키로 하고 판매 내역을 vector<pair<int(bid), int(num)>>에 저장
unordered_map<int, vector<pii>> sTab;
```

Solution sketch

- 구매 아이디 별 정보를 다음과 같은 클래스를 이용하여 저장할 수 있다.

```
struct Product {
    // 구입id, 상품id, 가격, 구입 수, 재고 수
    int Bid, Pid, price, num, snum;
    int cancel(int pid) {
        if (num != snum) return -1;    // 원래 수량이 아닌 경우
        prodSum[pid] -= num, snum = 0;// 취소 처리(팔 수 있는 물건이 없음)
        return prodSum[pid];
    }
}prod[BLM];
```

Solution sketch

- 상품 번호 별 우선순위 자료구조를 위하여 다음과 같은 자료 구조를 사용할 수 있다.

```
struct Comp {
    bool operator()(const int a, const int b) const {
        if (prod[a].price != prod[b].price)
            return prod[a].price > prod[b].price;
        return prod[a].Bid > prod[b].Bid;
    }
};

// pid별로 bid를 우선순위에 맞게 관리
priority_queue<int, vector<int>, Comp> pq[PLM];
```

- 우선순위 큐에서는 bid만을 Comp 함수 객체를 이용하여 관리하고(이를 포인터로 대신할 수 있다.) 실제 데이터는 prod[bid]의 멤버들을 통해 관리된다. 제품별 개수는 prodSum[PLM]을 통해 관리된다.

Solution sketch

이제 준비된 자료구조를 이용하여 각 함수 별 할 일을 정리해 보자.

`void init() :`

- 우선순위 큐와 prodSum[]을 초기화 한다. : pq, prodSum[]
- 구입번호, 제품번호 renumbering id 를 초기화 한다. : bCnt, pCnt
- 구입번호, 제품번호 renumbering id 를 위한 hash table을 초기화 한다. : bHash, pHash
- 판매 내역을 초기화 한다. : sTab

Solution sketch

```
int buy(int BID, int product, int price, int quantity) :
```

- 판매 아이디를 renumbering 한 bid를 얻는다.
- 상품 번호를 renumbering 한 pid를 얻는다.
- prod[bid] = {BID, product, price, quantity, quantity} 를 등록한다.
- 우선순위 큐 pq[pid]에 bid를 추가한다.
- 상품별 재고 관리 테이블 prodSum[pid]에 quantity를 더해 준다.
- prodSum[pid]를 반환한다.

Solution sketch

int cancel(int BID) :

- BID가 존재하지 않는 다면 -1을 반환하고 함수를 종료한다.
- 판매 아이디를 renumbering 한 bid를 얻는다.
- 상품 번호를 renumbering 한 pid를 얻는다.
- prod[bid]의 멤버 함수를 이용하여 취소 결과를 반환한다.

```
int prodSum[PLM];
struct Product {
    // 구입id, 상품id, 가격, 구입수, 재고수
    int Bid, Pid, price, num, snum;
    int cancel(int pid) {
        if (num != snum) return -1;     // 원래 수량이 모두 남아있지 않은 경우
        prodSum[pid] -= num, snum = 0; // 취소 처리(팔수 있는 물건이 없음)
        return prodSum[pid];
    }
}prod[BLM];
```

Solution sketch

`int sell(int SOD, int product, int price, int quantity) :`

- product 제품이 입고된 적이 없는 경우 -1을 반환하고 함수를 종료한다.
- 상품 번호를 renumbering 한 pid를 얻는다.
- pid제품의 재고가 부족한 경우 -1을 반환하고 함수를 종료한다.
- 그렇지 않은 경우 아래와 같은 방법으로 res를 구하여 반환하고 함수를 종료한다.

```
int res = 0;
while (!pq[pid].empty() && quantity) {
    int bid = pq[pid].top();
    if (prod[bid].snum == 0) { pq[pid].pop(); continue; }
    int minNum = min(prod[bid].snum, quantity);// 판매 가능한 개수
    prod[bid].snum -= minNum;                  // 재고 개수 업데이트
    quantity -= minNum;                        // 구매해야 할 개수 업데이트
    prodSum[pid] -= minNum;                    // pid제품 개수 업데이트
    res += (price - prod[bid].price) * minNum; // 판매 수익 구하기
    sTab[SID].push_back({bid, minNum});        // 취소를 위하여 판매 기록 보관
}
```

Solution sketch

int refund(int SID) :

- 환불에 실패한 경우(판매 내역이 없거나 이미 환불한 경우)
  -1을 반환하고 함수를 종료한다.
- SID판매 내역인 sTab[SID]를 순회하며 다음 프로세스를 진행한다.

```
int res = 0;
for (auto&p : sTab[SID]) {
    int bid = p.first, snum = p.second;
    int pid = pHash[prod[bid].Pid];
    if (prod[bid].snum == 0) pq[pid].push(bid);
    prod[bid].snum += snum;
    res += snum;
    prodSum[pid] += snum;
}
sTab.erase(SID);
```

- res를 반환하고 함수를 종료한다.

Solution sketch

[Summary]

- 배열의 인덱스로 다루기 어려운 큰 정수를 renumbering하는데 unordered_map, hash, trie등의 자료구조를 사용할 수 있다.

- 우선순위 자료구조를 위하여 함수 객체를 사용할 수 있다.

- 우선순위 자료구조를 위하여 priority_queue, heap등을 사용할 수 있다.

Code example

Code example

```
#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#endif

#include <cstdio>
#include <vector>
#include <queue>
#include <algorithm>
#include <unordered_map>
using namespace std;

using pii = pair<int, int>;
const int PLM = 605;
const int BLM = 30005;
int prodSum[PLM];
struct Product {
    int Bid, Pid, price, num, snum;    // 구입id, 상품id, 가격, 구입 수, 재고 수
    int cancel(int pid) {
        if (num != snum) return -1;    // 원래 수량이 모두 남아있지 않은 경우
        prodSum[pid] -= num, snum = 0; // 취소 처리(팔수 있는 물건이 없음)
        return prodSum[pid];
    }
}prod[BLM];
```

Code example

```
struct Comp {
    bool operator()(const int a, const int b) const {
        if (prod[a].price != prod[b].price) return prod[a].price > prod[b].price;
        return prod[a].Bid > prod[b].Bid;
    }
};

priority_queue<int, vector<int>, Comp> pq[PLM];  // pid를 우선순위에 맞게 관리
unordered_map<int, int> bHash, pHash;            // 구입번호, 제품번호 renumbering id를 위
한 hash table
int bCnt, pCnt;                                  // 구입번호, 제품번호 renumbering id
unordered_map<int, vector<pii>> sTab;            // SID를 키로 판매 내역을 vectro<pii>에 저장

void init() {
    for (int i = 0; i < PLM; ++i) pq[i] = {}, prodSum[i] = 0;
    bCnt = pCnt = 0;
    bHash.clear(), pHash.clear(), sTab.clear();
}
```

Code example

```
int buy(int BID, int product, int price, int quantity) {
    int bid = bHash.count(BID) ? bHash[BID] : bHash[BID] = ++bCnt;
    int pid = pHash.count(product) ? pHash[product] : pHash[product] = ++pCnt;
    prod[bid] = { BID, product, price, quantity, quantity };
    pq[pid].push(bid);
    prodSum[pid] += quantity;
    return prodSum[pid];
}

int cancel(int BID) {
    if (bHash.count(BID) == 0) return -1;
    int bid = bHash[BID];
    int pid = pHash[prod[bid].Pid];
    return prod[bid].cancel(pid);
}
```

Code example

```
int sell(int SID, int product, int price, int quantity) {
    if (pHash.count(product) == 0) return -1;       // procuct 제품이 입고된 적이 없는 경우
    int pid = pHash[product];
    if (prodSum[pid] < quantity) return -1;         // 재고가 부족한 경우
    int res = 0;
    while (!pq[pid].empty() && quantity) {
        int bid = pq[pid].top();
        if (prod[bid].snum == 0) {
            pq[pid].pop(); continue;
        }
        int minNum = min(prod[bid].snum, quantity);// 판매 가능한 개수
        prod[bid].snum -= minNum;                  // 판매 개수 업데이트
        quantity -= minNum;                        // 구매해야 할 개수 업데이트
        prodSum[pid] -= minNum;                    //
        res += (price - prod[bid].price) * minNum; // 판매 수익 구하기
        sTab[SID].push_back({bid, minNum});        // 취소를 위하여 판매 기록 보관
    }
    return res;
}
```

Code example

```
int refund(int SID) {
    if (sTab.count(SID) == 0) return -1;  // 환불에 실패한 경우(구매내역 없거나 이미환불)
    int res = 0;
    for (auto&p : sTab[SID]) {
        int bid = p.first, snum = p.second;
        int pid = pHash[prod[bid].Pid];
        if (prod[bid].snum == 0) pq[pid].push(bid);
        prod[bid].snum += snum;
        res += snum;
        prodSum[pid] += snum;
    }
    sTab.erase(SID);
    return res;
}
```

Thank you.