

삼성청년 SW·AI아카데미

SWEA2477 차량정비소

차량 정비소

목표 : 지갑을 분실한 고객과 같은 접수 창구와 같은 정비 창구를 이용한 고객의 고객번호들을 찾아 그 합을 출력하는 프로그램을 작성하라.

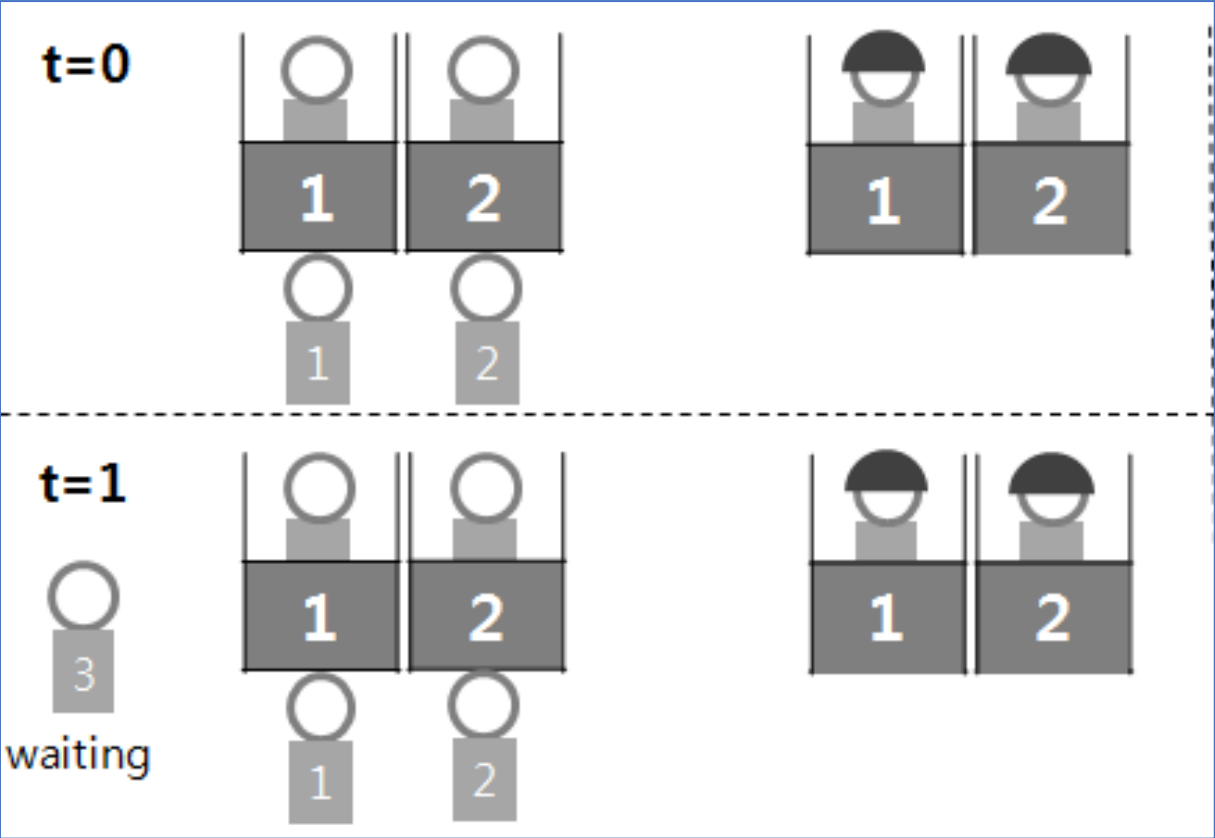
- 시뮬레이션

규칙

1. 정비 창구와 접수 창구가 있다. (각각 N, M 개)
2. 고객은 도착하는 대로 1번부터 번호가 부여된다.
3. 접수 창구 → 정비 창구 순으로 이동
4. 각 창구의 우선 처리 방식이 다르다.

1. 시간제한 : 최대 50개 테스트 케이스를 모두 통과하는데, C/C++/Java 모두 3초
2. 접수 창구의 개수 N , 정비 창구의 개수 M ($1 \leq N, M \leq 9$)
3. 각 접수 창구에서 처리하는 시간 a_i ($1 \leq a_i \leq 20$)
4. 각 정비 창구에서 처리 시간 b_j ($1 \leq b_j \leq 20$)
5. 방문한 고객의 수 K ($2 \leq K \leq 1,000$)
6. 고객의 도착 시간 t_k ($0 \leq t_k \leq 1,000$)
7. 지갑을 두고 간 고객이 이용한 접수 창구번호 A ($1 \leq A \leq N$)
8. 지갑을 두고 간 고객이 이용한 정비 창구번호 B ($1 \leq B \leq M$)
9. 창구번호와 고객번호는 1부터 시작한다.

천천히 예시를 이해한다.



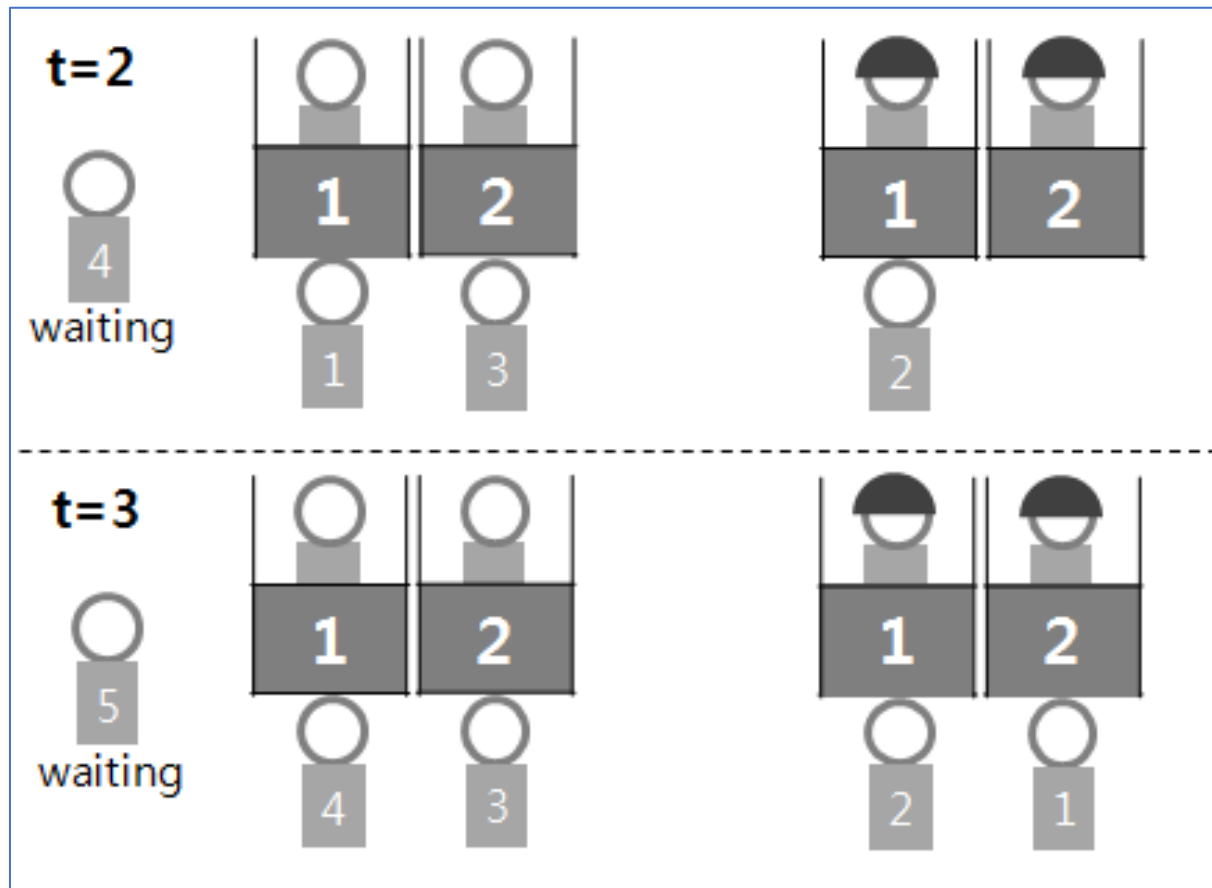
접수 창구번호	접수 창구의 처리 시간
1	$a_1 = 3$
2	$a_2 = 2$
정비 창구번호	정비 창구의 처리 시간
1	$b_1 = 4$
2	$b_2 = 2$

[Table 1]

고객번호	고객의 도착 시간
1	$t_1 = 0$
2	$t_2 = 0$
3	$t_3 = 1$
4	$t_4 = 2$
5	$t_5 = 3$
6	$t_6 = 4$

[Table 2]

천천히 예시를 이해한다.



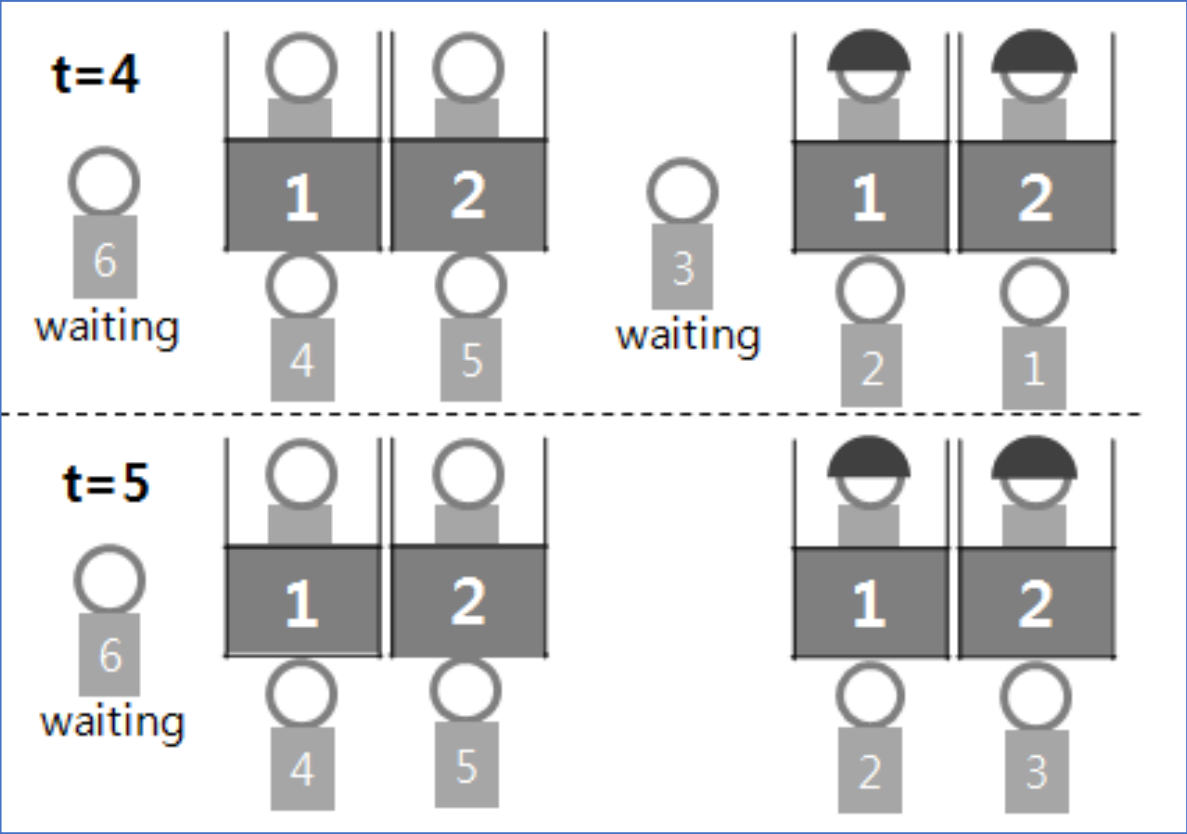
접수 창구번호	접수 창구의 처리 시간
1	$a_1 = 3$
2	$a_2 = 2$
정비 창구번호	정비 창구의 처리 시간
1	$b_1 = 4$
2	$b_2 = 2$

[Table 1]

고객번호	고객의 도착 시간
1	$t_1 = 0$
2	$t_2 = 0$
3	$t_3 = 1$
4	$t_4 = 2$
5	$t_5 = 3$
6	$t_6 = 4$

[Table 2]

천천히 예시를 이해한다.



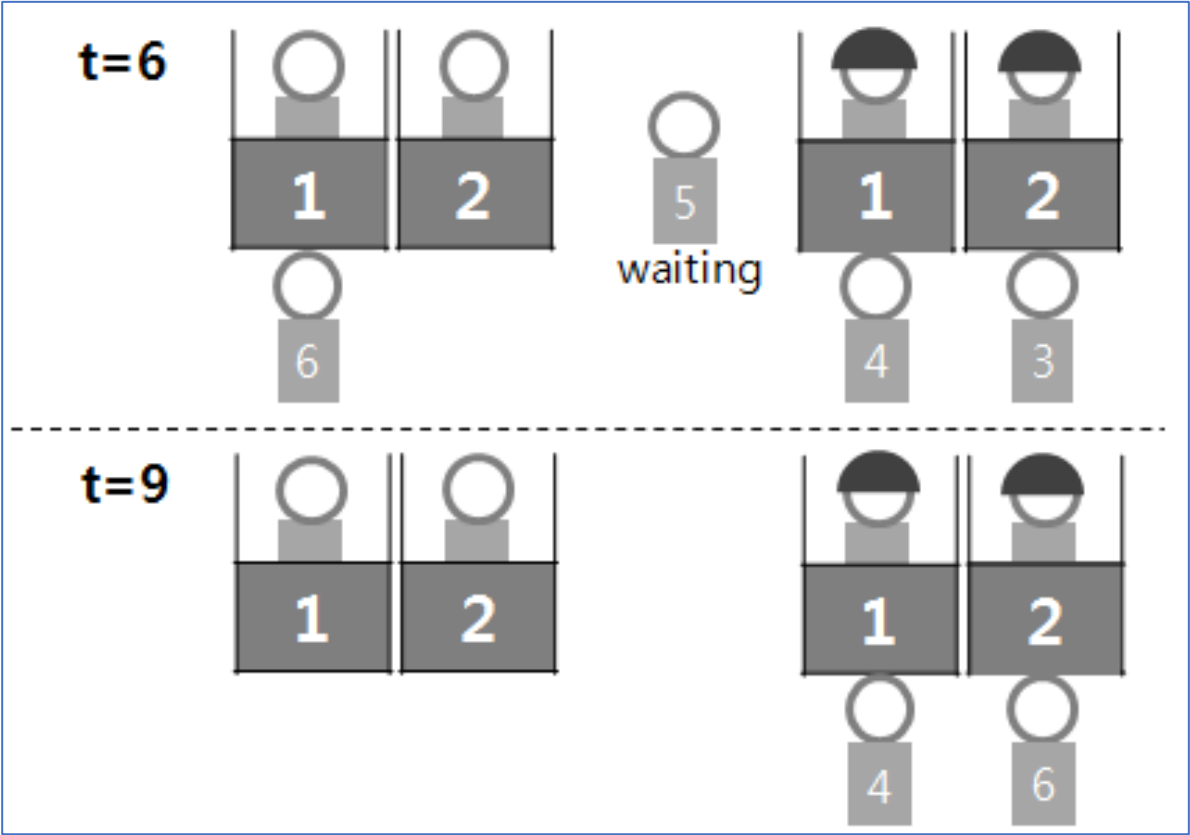
접수 창구번호	접수 창구의 처리 시간
1	$a_1 = 3$
2	$a_2 = 2$
정비 창구번호	정비 창구의 처리 시간
1	$b_1 = 4$
2	$b_2 = 2$

[Table 1]

고객번호	고객의 도착 시간
1	$t_1 = 0$
2	$t_2 = 0$
3	$t_3 = 1$
4	$t_4 = 2$
5	$t_5 = 3$
6	$t_6 = 4$

[Table 2]

천천히 예시를 이해한다.



접수 창구번호	접수 창구의 처리 시간
1	$a_1 = 3$
2	$a_2 = 2$
정비 창구번호	정비 창구의 처리 시간
1	$b_1 = 4$
2	$b_2 = 2$

[Table 1]

고객번호	고객의 도착 시간
1	$t_1 = 0$
2	$t_2 = 0$
3	$t_3 = 1$
4	$t_4 = 2$
5	$t_5 = 3$
6	$t_6 = 4$

[Table 2]

데이터를 입력 받는 것이 가장 중요하다.

- 창구 번호와 고객 번호가 1부터 넘버링이 된다.

```
6 //접수 창구 개수 N
7 //정비 창구 개수 M
8 //고객 수 K
9 int N, M, K;
10
11 //지갑을 두고간 고객이 이용한 접수 창구 번호
12 int A;
13 //지갑을 두고간 고객이 이용한 정비 창구 번호
14 int B;
15
16 //접수 창구 별 처리 시간 배열
17 int ai[11];
18 //정비 창구 별 처리 시간 배열
19 int bj[11];
```

```
//테스트 케이스
int T;
cin >> T;

for (int tc = 1; tc <= T; tc++) {
    cin >> N >> M >> K >> A >> B;

    //창구번호와 고객번호는 1부터
    for (int i = 1; i <= N; i++) cin >> ai[i];
    for (int i = 1; i <= M; i++) cin >> bj[i];
    for (int i = 1; i <= K; i++) cin >> tk[i];

    //결과 출력
    int result = 0;
    cout << "#" << tc << " " << result << '\n';
}
```

고객마다 어떤 접수 창구와 정비 창구를 방문 했는 지 파악하기 위해
고객 정보를 다룰 구조체를 만들 필요가 있다.

- 해당 고객이 도착한 시간(tk)도 고객 정보이므로 묶어서 처리한다.

```
for (int i = 1; i <= N; i++) cin >> ai[i];
for (int i = 1; i <= M; i++) cin >> bj[i];
//고객이 도착한 시간 배열 tk 를 구조체에 담아서 처리
for (int i = 1; i <= K; i++) cin >> man[i].arrive_time;
```

```
struct NODE {
    //고객이 방문한 접수창구번호
    int aNum;
    //고객이 방문한 정비창구번호
    int bNum;
    //고객이 도착한 시간(tk)
    int arrive_time;
};
//고객 정보를 담은 구조체 배열
NODE man[1001];
```

시뮬레이션 후

고객마다 방문한 접수창구 번호와 정비창구 번호를

미리 입력 받은 A,B 값과 비교해서 있으면 고객 번호의 합산 값을
없으면 -1을 출력한다.

```
//결과 출력
int result = 0;

for (int i = 1; i <= K; i++) {
    if (man[i].aNum == A && man[i].bNum == B) {
        result += i;
    }
}

if (result > 0) cout << "#" << tc << " " << result << '\n';
else    cout << "#" << tc << " " << -1 << '\n';
}
```

시뮬레이션 구현하기

1. 1~K명의 고객들이 도착하는 대로 접수창구로 이동
2. 1~N개의 접수 창구에서 작업이 끝난 고객을 정비 창구로 이동
3. 1~N개의 접수 창구 중 비어있는 창구 있으면 대기하는 고객 넣기
 - 비어있는 지 여부 체크
4. 1~M개의 정비 창구에서 작업이 끝난 고객은 정비소에서 떠난다.
5. 1~M개의 정비 창구 중 비어 있는 창구 있으면 대기하는 고객 넣기
 - 비어있는 지 여부 체크
6. 정비소의 작업이 모두 끝났다면 종료
7. 시간 흐름

접수 창구, 정비 창구용 큐를 만든다.

- **접수 창구의 우선순위**
 - 고객번호가 낮은 순서대로 우선 접수
 - 빈 창구가 여러 곳인 경우, 창구 번호가 작은 곳으로 간다.
- **정비 창구의 우선순위**
 - 먼저 기다리는 고객 우선
 - 접수 창구에서 여러 명이 이동할 때, 낮은 번호의 접수창구번호의 고객부터 우선된다
 - 빈 창구가 여러 곳인 경우, 창구 번호가 작은 곳으로 간다.

```
30 //접수 창구 대기용 큐
31 queue<int> aQueue;
32 //정비 창구 대기용 큐
33 queue<int> bQueue;
```

- 접수 창구와 정비 창구를 이용하는 고객 번호를 알아야 한다.

- 각 창구에서 작업이 끝난 뒤, 이동을 추적해야 함
- 창구 길이 크기의 배열 생성

```
//접수 창구별 이용하는 고객번호  
int aPerson[11];  
//정비 창구별 이용하는 고객번호  
int bPerson[11];
```

- 각 창구 별 종료 시간도 기록해야 한다.

- 시뮬레이션을 돌리면서 해당 시간이 되면, 창구에서의 작업이 끝나서 다음 창구로 이동하거나 종료된다.
- 종료되는 시점도 알기 위해서 종료 될 때마다 카운팅 작업 필요

```
//접수 창구별 종료 시간  
int aEndTime[11];  
//정비 창구별 종료 시간  
int bEndTime[11];
```

내일 방송에서 만나요!

삼성청년SW·AI아카데미