〈자료구조 실습〉 - 집합

※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↦ 이 후는 각 입력과 출력에 대한 설명이다.

[문제 1] 두 개의 집합 A와 B를 입력 받아, A가 B의 부분집합인지를 검사하는 프로그램을 작성하시오.

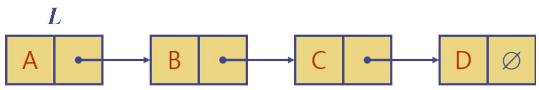
주의:

1) 집합은 오름차순 양의 정수로 저장 및 출력되어야 한다.
2) 공집합은 공집합을 포함한 모든 집합의 부분집합이다.
3) **입력:** 프로그램은 두 개의 집합 A, B를 차례로 표준입력 받는다. 한 개의 집합을 나타내는 두 개의 입력 라인은 다음과 같이 구성된다.

   첫 번째 라인: 정수 n (집합 크기, 즉 집합 원소의 개수)

   두 번째 라인: 집합의 원소들 (오름차순 양의 정수 수열).

   공집합은 첫 번째 라인은 0, 두 번째 라인은 존재하지 않는다.
4) **출력:** A ⊂ B이면 0을 출력하고, 그렇지 않으면 집합 B에 속하지 않은 집합 A의 가장 작은 원소를 표준 출력한다.
5) 모든 집합은 헤더 노드가 없는 단일연결리스트(singly-inked list) 형태로 구축되어야 한다.
6) **참고:** 아래 그림은 일반적인 단일연결리스트를 나타낸다. 빈 리스트의 경우 null pointer로 나타낸다. (그림의 노드에 저장된 원소가 영문자인데, 이는 무시하고 리스트의 형태만 참고하시오.)



입력 예시 1

```
3                ↦ 집합 A 크기
4 6 13           ↦ 집합 A
6                ↦ 집합 B 크기
1 3 4 6 8 13     ↦ 집합 B
```

출력 예시 1

```
0        ↦ A ⊂ B
```

입력 예시 2

```
3                ↦ 집합 A 크기
7 10 53          ↦ 집합 A
4                ↦ 집합 B 크기
7 10 15 45       ↦ 집합 B
```

출력 예시 2

```
53       ↦ A ⊄ B
```

입력 예시 3

0	↦ 집합 A (공집합)
3	↦ 집합 B 크기
9 20 77	↦ 집합 B

출력 예시 3

0	↦ A ⊂ B

입력 예시 4

0	↦ 집합 A (공집합)
0	↦ 집합 B (공집합)

출력 예시 4

0	↦ A ⊂ B

다음 함수를 작성하여 사용하시오.

- 함수 subset : 집합 A가 집합 B의 부분집합인지 여부 검사
  - 인자: 양의 정수 집합 A, B (A, B는 각각 단일연결리스트의 헤드 노드)
  - 반환값: 정수 (A ⊂ B면 0, 그렇지 않으면 집합 B에 속하지 않은 집합 A의 가장 작은 원소)

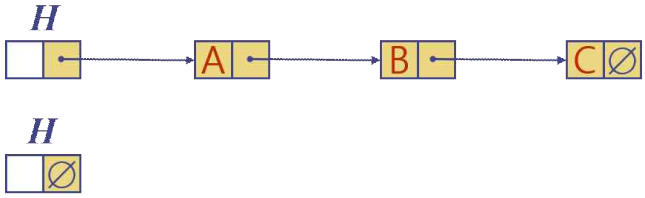
- void AppendNode(Set **List, int element); 함수 원형을 지켜서 집합을 생성.

- 단일 연결 리스트 이고 헤더가 없는 단일 리스트로 구성. (빈 헤더가 없이 곧장 A, B 포인터가 리스트를 가리키도록 , 별도의 헤더 구조체를 유지하지 말 것.)

[문제 2] 두 개의 집합을 입력받아, 합집합과 교집합을 구하는 프로그램을 작성하시오.

주의:

1) 모든 집합은 오름차순 양의 정수로 저장 및 출력되어야 한다.
2) 공집합 처리에 주의
3) **입력:** 프로그램은 두 개의 집합 A, B를 차례로 표준입력 받는다. 한 개의 집합을 나타내는 두 개의 입력 라인은 다음과 같이 구성된다.
   첫 번째 라인: 정수 n (집합 크기, 즉 집합 원소의 개수)
   두 번째 라인: 집합의 원소들 (오름차순 양의 정수 수열).
   따라서 공집합은 첫 번째 라인은 0, 두 번째 라인은 존재하지 않는다.
4) **출력:** 각 연산 결과는 두 개의 라인으로 표준출력한다. 첫 번째 라인은 합집합을, 두 번째 라인은 교집합을 나타낸다. 이때 공집합은 0로 출력한다.
5) 모든 집합은 헤더(header) 노드가 추가된 단일연결리스트 형태로 구축되어야 한다.
6) **참고:** 아래 첫 번째 그림은 일반적인 헤더 단일연결리스트를, 아래 두 번째 그림은 빈 리스트를 나타낸다. (그림의 노드에 저장된 원소가 영문자인데, 이는 무시하고 리스트의 형태만 참고하시오.)



입력 예시 1

```
6                   ↦ 집합 A 크기
3 7 45 88 99 101    ↦ 집합 A
4                   ↦ 집합 B 크기
7 10 15 45          ↦ 집합 B
```

출력 예시 1

```
□3 7 10 15 45 88 99 101    ↦ 합집합
□7 45                      ↦ 교집합
```

입력 예시 2

```
0                ↦ 집합 A 크기 (공집합)
3                ↦ 집합 B 크기
9 20 77          ↦ 집합 B
```

출력 예시 2

```
□9 20 77         ↦ 합집합
□0               ↦ 교집합 (공집합)
```

입력 예시 3

```
0                ↦ 집합 A 크기 (공집합)
0                ↦ 집합 B 크기 (공집합)
```

출력 예시 3

```
□0               ↦ 합집합 (공집합)
□0               ↦ 교집합 (공집합)
```

다음 함수를 작성하여 사용하시오.

- 함수 union: 합집합 연산
  - 인자: 양의 정수 집합 A, B (A, B는 각각 헤더 단일연결리스트의 헤더 노드)
  - 반환값: A∪B의 헤더 노드 주소 또는, 공집합인 경우 빈 리스트(즉, 헤더 노드만 존재)
- 함수 intersect: 교집합 연산
  - 인자: 양의 정수 집합 A, B (A, B는 각각 헤더 단일연결리스트의 헤더 노드)
  - 반환값: A∩B의 헤더 노드 주소 또는, 공집합인 경우 빈 리스트(즉, 헤더 노드만 존재)

- void AppendNode(Set *Header, int element);

// 헤더가 없는 1의 경우와 비교

- Set *union(Set *Header1, Set *Header2)

- Set *intersect(Set *Header1, Set *Header2)

- Set1, Set2와 결과 Set은 모두 헤더가 있는 단일연결 리스트로 구성하고

헤더의 정보는 내용이 아무것도 없는 더미 헤더로 사용(리스트에 대한 정보를 담고 있으면 안 됨.)