Unscrambling a Messy Bug

일샤트는 효율적인 자료 구조를 설계하는 소프트웨어 엔지니어이다. 어느날 그는 새로운 자료 구조를 고안해냈다. 이 자료 구조는 n 비트 길이의 **음이 아닌** 정수 집합을 저장한다. 여기에서 n 은 2의 자승이다. 즉, $n = 2^b$ 인 어떤 음이 아닌 정수 b 가 존재한다.

이 자료 구조는 처음에는 비어 있다. 이 자료 구조를 사용하는 프로그램은 다음 규칙을 따라야 한다.

- 프로그램은 n -비트 정수인 원소들을 `add_element(x)` 함수를 사용하여 한 번에 하나씩 자료 구조에 저장할 수 있다. 만약 프로그램이 자료 구조에 이미 포함되어 있는 원소를 삽입하려고 하면, 아무 일도 일어나지 않는다.
- 마지막 원소를 삽입한 다음에는 프로그램은 함수 `compile_set()` 함수를 정확하게 한 번 호출한다.
- 마지막으로, 프로그램은 `check_element(x)` 함수를 사용하여 원소 x 가 자료 구조에 저장되어 있는지 확인할 수 있다. 이 함수는 여러 번 사용될 수 있다.

일샤트가 처음 자료 구조를 구현하였을 때, `compile_set()` 함수의 구현이 잘못되어 버그가 있었다. 이 버그는 집합에 저장된 모든 원소들에 대해서 원소를 이진수로 표현했을때 똑같은 방식으로 비트들의 순서를 바꾼다. 일샤트는 이 버그때문에 어떤 식으로 비트들이 바뀌는지 당신이 정확하게 알려주었으면 한다.

보다 엄밀하게, 0 부터 $n-1$ 까지 모든 숫자가 정확하게 한 번씩 나오는 수열 $p = [p_0, \ldots, p_{n-1}]$ 을 생각해보자. 우리는 이 수열을 **순열(permutation)**이라고 한다. 집합에 포함되어 있는 원소 하나에 대해서, 이 원소를 이진수로 표현했을 때 각 비트가 $a_0, \ldots, a_{n-1}$ 이라고 하자. (a_0 이 최상위 비트이다.) `compile_set()` 함수가 호출되었을 때, 이 원소의 값은 $a_{p_0}, a_{p_1}, \ldots, a_{p_{n-1}}$ 로 바뀐다.

동일한 순열 p 가 집합의 모든 원소들의 비트 순서를 바꾸는데 사용된다. 이 순열은 어떤 순열도 가능하다. 모든 $0 \le i \le n-1$ 에 대해서 $p_i = i$ 가 되는 경우도 가능하다.

예를 들어서, $n = 4, p = [2, 1, 3, 0]$ 인 경우를 생각해보자. 당신은 집합에 이진수로 표현했을 때 `0000`, `1100`, `0111`인 정수들을 삽입하였다. `compile_set` 함수가 호출되면, 이 원소의 값들은 각각 `0000`, `0101`, `1110`으로 바뀐다.

당신은 자료 구조와 상호 소통을 통하여 순열 p 를 찾는 프로그램을 짜야 한다. 프로그램은 다음 순서대로 진행해야 한다.

1. n -비트 정수들의 집합을 선택한다.
2. 이 정수들을 자료 구조에 저장한다.
3. `compile_set` 함수를 호출하여 버그가 동작하게 한다.
4. 어떤 원소가 바뀐 집합에 있는지 확인한다.
5. 이 정보를 통하여 순열 p 를 구하여 리턴한다.

당신의 프로그램이 `compile_set` 함수를 정확하게 한 번 호출해야 한다는데 유의하라.

추가로, 당신의 프로그램이 라이브러리 함수를 호출하는 횟수에 제한이 있다.
- `add_element` 함수는 최대 w 번 호출될 수 있다. (w 는 "write"의 첫글자)
- `check_element` 함수는 최대 r 번 호출될 수 있다. (r 는 "read"의 첫글자)

Implementation details

하나의 함수(혹은 method)를 작성한다:
- `int[] restore_permutation(int n, int w, int r)`
  - `n`: 집합의 각 원소들을 이진수로 표현했을 때 비트 수 (동시에, p 의 길이)
  - `w`: 당신의 프로그램이 `add_element` 함수를 호출할 수 있는 최대 횟수.
  - `r`: 당신의 프로그램이 `check_element` 함수를 호출할 수 있는 최대 횟수.
  - 함수의 리턴 값은 순열 p 이다.

C 언어인 경우 아래와 같이 작성한다:
- `void restore_permutation(int n, int w, int r, int* result)`
  - `n`, `w`, `r`의 의미는 위와 동일하다.
  - 이 함수는 파라미터로 주어진 배열 `result`에 순열 p 를 저장한다. 각각의 i 에 대해, p_i 는 `result[i]`에 저장해야 한다.

Library functions

자료 구조와 상호 소통을 위해서 당신의 프로그램은 다음 세 함수를 사용해야 한다.
- `void add_element(string x)`
  이 함수는 `x`로 표현되는 원소를 집합에 추가한다.
  - `x`: `'0'`과 `'1'`로 이루어진 문자열인데, 이 집합에 들어갈 정수를 이진수로 표현한 것이다. `x`의 길이는 n 이어야 한다.

- `void compile_set()`
  이 함수는 정확하게 한 번 호출된다. 당신의 프로그램은 이 함수를 호출한 후에는 `add_element()` 함수를 호출할 수 없다. 또한, 이 함수를 호출하기 전에 `check_element()` 함수를 호출할 수 없다.
- `boolean check_element(string x)`
  이 함수는 원소 `x`가 버그의 영향을 받은 집합에 포함되는지를 점검한다.
  - `x`: `'0'`과 `'1'`로 이루어진 문자열인데, 이 집합에 포함되는지 점검할 정수를 이진수로 표현한 것이다. `x`의 길이는 n 이어야 한다.
  - `x`가 버그의 영향을 받은 집합에 포함되어 있다면 리턴 값은 `true`이고, 그렇지 않다면 `false`이다.

만약 여러분의 프로그램이 위 제약 조건을 지키지 않았다면, 채점 결과는 "Wrong Answer"이다.

사용되는 모든 문자열은, 이 문자열의 첫 글자가 해당하는 정수의 최상위 비트이다.

그레이더는 함수 `restore_permutation`를 호출하기 전에 순열 p 를 확정한다.

제공되는 템플리트 파일에 당신이 사용하는 프로그래밍 언어에서 구현하는데 필요한 세부 사항이 들어 있다.

Example

그레이더가 다음 함수 호출을 한다.
- `restore_permutation(4, 16, 16)`. $n = 4$ 이고, 이 프로그램은 최대 16 번 "읽고",

최대 16 번 "쓸 수" 있다.

프로그램은 다음 순서로 함수들을 호출한다.

- `add_element("0001")`
- `add_element("0011")`
- `add_element("0100")`
- `compile_set()`
- `check_element("0001")` 의 리턴값은 `false`
- `check_element("0010")` 의 리턴값은 `true`
- `check_element("0100")` 의 리턴값은 `true`
- `check_element("1000")` 의 리턴값은 `false`
- `check_element("0011")` 의 리턴값은 `false`
- `check_element("0101")` 의 리턴값은 `false`
- `check_element("1001")` 의 리턴값은 `false`
- `check_element("0110")` 의 리턴값은 `false`
- `check_element("1010")` 의 리턴값은 `true`
- `check_element("1100")` 의 리턴값은 `false`

`check_element()` 함수의 호출로부터 얻을 수 있는 결과와 일치하는 순열은 오직 하나 존재하는데, 순열 $p = [2, 1, 3, 0]$ 이다. 따라서, `restore_permutation`의 리턴 값은 `[2, 1, 3, 0]`이어야 한다.

Subtasks

1. (20 points) $n = 8$, $w = 256$, $r = 256$, 최대 두 곳의 인덱스 i 에서 $p_i \neq i$ ($0 \leq i \leq n-1$),
2. (18 points) $n = 32$, $w = 320$, $r = 1024$,
3. (11 points) $n = 32$, $w = 1024$, $r = 320$,
4. (21 points) $n = 128$, $w = 1792$, $r = 1792$,
5. (30 points) $n = 128$, $w = 896$, $r = 896$.

Sample grader

sample grader는 다음의 형식으로 입력을 읽어들인다:

- line 1: 정수 n, w, r.
- line 2: p 의 원소를 나타내는 n 개의 정수.