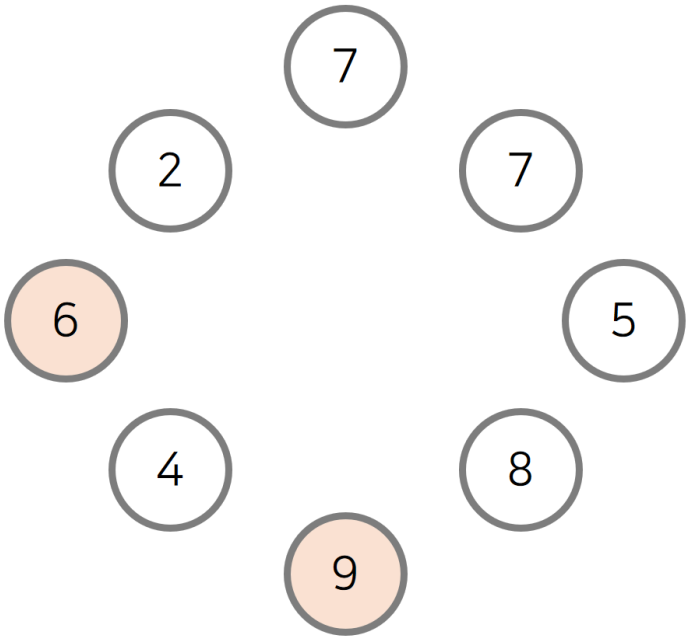
# 자신감 점수  문제

$N$명의 전사가 원형으로 배치되어 있다. 이때, 전사들은 자신과 인접한 두 전사의 전투력을 확인할 수 있다. 만약 인접한 두 전사의 전투력이 모두 자신의 전투력보다 낮다면, 해당 전사는 자신감이 있는 상태가 된다. 반면에 인접한 두 전사 중에서 한 명이라도 자신의 전투력과 같거나 혹은 높다면, 해당 전사는 자신감이 없는 상태가 된다.

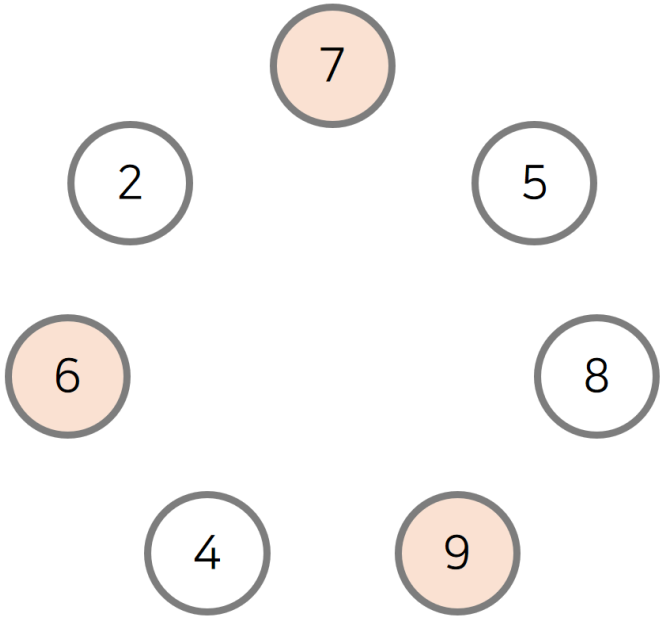
모든 전사들이 원형으로 배치되어 있는 상태가 주어졌을 때, 자신감이 있는 상태인 전사의 수를 **자신감 점수**라고 하자.

$N$명의 전사를 이끄는 장군은 기존의 배치를 유지한 상태로 한 명의 전사를 열외시켜야 한다. 한 명의 전사를 열외시킴으로써 얻을 수 있는 자신감 점수의 최댓값을 구하는 프로그램을 계산하여라.

예를 들어 다음과 같이 $N = 8$명의 전사의 전투력 정보가 [7, 7, 5, 8, 9, 4, 6, 2] 형태로 주어졌다고 가정하자. 이는 다음과 같이 원형으로 배치되어 있는 것을 의미한다. 각 수의 의미는 해당 전사의 전투력 값을 의미한다. 현재 배치에서 **자신감 점수**는 2점이다. 자신감이 있는 상태인 전사를 빨간색으로 음영 처리 하였다.




이때, 2번째 전사를 열외시키는 경우 다음과 같은 상태가 되며, **자신감 점수**는 3점이 된다.




따라서, 현재 예시에서는 정답이 3이다.

## 입력 조건

가장 먼저 전사의 수 $N$이 자연수로 주어진다. ($N$은 4 이상 $1,000$ 이하의 자연수다.)
이어서 1번부터 $N$번까지의 전사가 초기에 원형으로 서 있는 정보가 담긴 배열 $warriors$가 주어진다. 이때, 각 전투력 값은 100보다 작거나 같은 자연수다.

## 출력 조건

기존의 배치를 유지한 상태로 한 명의 전사를 열외시킴으로써 얻을 수 있는 자신감 점수의 최댓값을 자연수 형태로 반환하여라.

## 입출력 예시

| $N$ | $warriors$ | 정답 |
|---|---|---|
| 8 | [7, 7, 5, 8, 9, 4, 6, 2] | 3 |

| | | |
|---|---|---|
| 10 | [5, 3, 7, 5, 3, 3, 6, 1, 8, 7] | 3 |
| 12 | [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] | 1 |

## 해설. 자신감 점수

본 문제는 아래의 두 과정으로 해결할 수 있다.

① 자신감 점수 계산 함수 작성하기

먼저 $M$명의 전사에 대한 전투력 정보가 주어졌을 때, 자신감 점수를 계산하는 함수를 작성한다. 구체적으로 $M$명의 전사에 대한 전투력 정보가 배열 형태로 주어진 상황을 가정하자.

이후에 기존의 배열에서 가장 왼쪽에 있던 전사를 가장 오른쪽 위치에, 기존의 배열에서 가장 오른쪽에 있던 전사를 가장 왼쪽 위치에 삽입하자. 그러면 크기가 $M + 2$인 배열이 만들어진다. 예를 들어 $M = 7$로, 전투력 정보가 [7, 5, 8, 9, 4, 6, 2] 형태로 주어졌다면 결과적으로 만들어지는 배열은 [2, 7, 5, 8, 9, 4, 6, 2, 7]이다.

이때, $i$번째 전사의 전투력 정보를 $a_i$라고 할 때, 인덱스 1부터 $M$까지 각 위치에 대하여 다음의 조건을 만족하는 전사의 수를 계산할 수 있다.

$a_{i-1} < a_i > a_{i+1}$

② 각 전사를 열외시켜 보며, 자신감 점수 계산하기

초기 전사의 수 $N$이 최대 1,000의 값을 가지므로, 각 전사를 일일이 열외시킨 상황에서의 자신감 점수를 계산할 수 있다. 이러한 완전 탐색을 사용하는 경우 시간 복잡도는 $O(N^2)$이다. 다시 말해 문제에서 제시한 요구 사항을 2중 반복문을 이용하여 해결할 수 있으며, 본 문제는 **기초 문법 및 완전 탐색 유형**에 속한다.

- **Python3 정답 코드 예시**

```
def confidence(M, arr):
    cnt = 0 # 자신감 점수
    for i in range(M):
        left = i - 1 # 왼쪽 전사
        if left == -1: left = M - 1
        right = i + 1 # 오른쪽 전사
        if right == M: right = 0
        # 자신감이 있는 상태인 전사인 경우
        if arr[left] < arr[i] and arr[i] > arr[right]:
            cnt += 1 # 카운트
    return cnt

# 전체 문제의 개수(N)과 각 전사의 전투력 배열(warriors)을 입력받기
def solution(N, warriors):
    answer = -1
    for i in range(N):
        arr = warriors[:i] + warriors[i + 1:]
        answer = max(answer, confidence(N - 1, arr))
    return answer
```

- **Java 정답 코드 예시**

```
class Solution {
    public static int confidence(int M, int[] arr) {
        int cnt = 0; // 자신감 점수
        for (int i = 0; i < M; i++) {
            int left = i - 1; // 왼쪽 전사
            if (left == -1) left = M - 1;
            int right = i + 1; // 오른쪽 전사
            if (right == M) right = 0;
            // 자신감이 있는 상태인 전사인 경우
            if (arr[left] < arr[i] && arr[i] > arr[right]) {
                cnt += 1; // 카운트
            }
        }
        return cnt;
    }
```

```
    // 전체 문제의 개수(N)과 각 전사의 전투력 배열(warriors)을 입력받기
    public static int solution(int N, int[] warriors) {
        int answer = -1;
        for (int i = 0; i < N; i++) {
            // i번째 전사를 제외한 배열 구하기
            int[] arr = new int[N - 1];
            for (int j = 0; j < i; j++) {
                arr[j] = warriors[j];
            }
            for (int j = i + 1; j < N; j++) {
                arr[j - 1] = warriors[j];
            }
            answer = Math.max(answer, confidence(N - 1, arr));
        }
        return answer;
    }
}
```