**[제한 사항]**

|  |  |
| --- | --- |
| 시간 | 25개 테스트케이스를 합쳐서 C++ 의 경우 3초 / Java 의 경우 3초 |
| 메모리 | 힙, 정적 메모리 합쳐서 256MB 이내, 스택 메모리 1MB 이내 |

**※** 본 문제는 임직원의 S/W 문제해결 역량 향상을 위한 **Professional 실전형 연습문제**이며

함께 제공되는 **Testcase 또한 학습을 위한 연습용**입니다.

|  |
| --- |
| **①   C 또는 C++로 답안을 작성하시는 응시자께서는 검정시스템에 제출 시, Language 에서 C++ 를 선택하신 후 제출하시기 바랍니다.**  **②   Main 과 User Code 부분으로 구성되어 있습니다.**  **ㅇ   A.**Main         : 수정할 수 없는 코드이며, 채점 시 비 정상적인 답안 검출 등 평가를 위한 로직이 추가 될 수 있습니다.  **ㅇ   B.**User Code  : 실제 응시자가 작성해야 하는 코드이며, 제출 시에는 표준 입출력 함수가 포함되어 있으면 안 됩니다.  **③   Local PC 에서 프로그래밍 시 유의 사항**  **ㅇ   A.**2개의 파일을 생성하셔야 합니다. ( main.cpp / solution.cpp 또는 Solution.java / UserSolution.java 또는 main.py / solution.py )  **ㅇ   B.**Main 부분의 코드를 main.cpp 또는 Solution.java 또는 main.py 에 복사해서 사용하시기 바랍니다.  **ㅇ   C.**sample\_input.txt 를 사용하시기 위해서는 Main 부분의 코드 내에  **ㅇ   C.**표준 입력을 파일로 전환하는 코드 ( 주석처리 되어 있음 ) 의 주석을 풀어서 사용하시면 됩니다.  **ㅇ   D.**User Code 부분의 코드를 작성하신 후 서버에 제출하실 때,  **ㅇ   D.**디버깅을 위한 표준 입출력 함수를 모두 삭제 또는 주석 처리해 주셔야 합니다.  **④   문제 내에 제약조건을 모두 명시하지 않으므로 주어지는 코드를 분석하셔야 합니다.**  **⑤   코드는 개발 언어에 따라 상이할 수 있으므로, 작성할 언어를 기준으로 분석하셔야 합니다.** |

**[문제 설명]**

2N개의 카드가 있다. (2 ≤ N ≤ 50,000)

각 카드의 앞면에는 1부터 N까지의 숫자 중 하나가 적혀 있다.

페어는 동일한 숫자가 적힌 서로 다른 카드 한 쌍을 의미한다.

주어진 카드에서 동일한 숫자가 적힌 카드는 항상 2개가 존재한다.

따라서 모든 카드를 정확히 N 쌍의 페어로 나눌 수 있다.

모든 페어 찾기 게임은 아래와 같은 순서로 진행된다.

① 모든 카드가 뒤집힌 상태로 주어진다. 따라서 각 카드에 적힌 숫자는 알 수 없다.

② 나열된 카드 중 2개를 선택하고 각 카드에 적힌 숫자 차이의 상한 값을 추측하여 심판에게 알린다.

③ 심판은 선택한 두 카드에 적힌 숫자의 차이가 추측한 상한 값과 같거나 작은지 여부를 확인하여 알려준다.

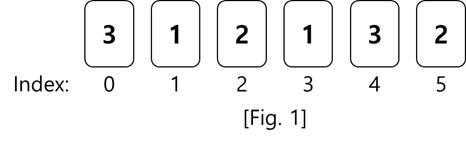
④ 만약 추측한 상한 값이 0이고 두 카드의 숫자가 같은 경우 해당 페어를 찾은 것으로 판정한다.

⑤ 아직 찾지 못한 페어가 있는 경우 ②번 과정부터 다시 진행한다.

⑥ 모든 페어를 찾으면 게임은 종료된다.

**[Fig. 1]** 은 6개의 카드로 모든 페어 찾기 게임을 시작했을 때의 상태를 간략하게 표기한 그림이다.

카드에 적힌 숫자는 순서대로 3, 1, 2, 1, 3, 2 이다.



**[Table. 1]** 은 두 개의 카드 위치와 적힌 숫자의 차이를 추측하여 정한 후 심판에게 질문했을 때의 결과를 나타낸다.

Answer은 심판이 알려주는 정보이며 Found는 찾은 페어를 나타낸다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | | | **Answer** | **Found** |
| **Index A** | **Index B** | **Difference** |
| 0 | 1 | 1 | FALSE | X |
| 0 | 2 | 1 | TRUE | X |
| 0 | 2 | 0 | FALSE | X |
| 0 | 3 | 1 | FALSE | X |
| 0 | 4 | 1 | TRUE | X |
| 0 | 4 | 0 | **TRUE** | **3** |
| 1 | 3 | 0 | **TRUE** | **1** |
| 2 | 5 | 0 | **TRUE** | **2** |

**[Table. 1]**

페어의 총 개수가 주어졌을 때 카드를 확인하는 작업을 여러 번 반복하여 모든 페어를 찾는 함수를 작성하라.

**아래 API 설명을 참조하여 모든 페어를 찾을 수 있도록 각 함수를 구현하라.**

※ 아래 함수 signature는 C/C++에 대한 것으로 다른 언어에 대해서는 제공되는 Main과 User Code를 참고하라.

아래는 User Code 부분에 작성해야 하는 API 의 설명이다.

|  |
| --- |
| **void playGame(int N)** |
| 각 테스트 케이스의 맨 처음에 호출된다.    N은 나열된 카드에 존재하는 서로 다른 페어의 개수이다.  전체 카드의 개수는 2N 이다.    각 카드에는 1부터 N까지 숫자 중 하나가 적혀 있다.  각 숫자가 적힌 카드는 2개씩 존재한다.    ***Parameters***  N : 서로 다른 페어의 개수 (2 ≤ N ≤ 50,000) |

아래는 User Code 에서 사용 가능한 Main API 의 설명이다.

|  |
| --- |
| **bool checkCards(int mIndexA, int mIndexB, int mDiff)** |
| 나열된 카드에서 2개의 카드를 고르고 각 카드에 적힌 숫자의 차이가 mDiff이하인지 확인한다.    mIndexA와 mIndexB는 각각 나열된 카드 중 선택할 두 카드의 위치를 의미한다.  mIndexA와 mIndexB는 0부터 2N-1까지의 숫자 중 하나를 선택한다.    mDiff가 0이고 서로 다른 mIndexA, mIndexB에 대해 두 위치의 카드에 적힌 숫자가 같은 경우  페어를 찾은 것으로 판단한다.    페어를 찾은 것과 관계없이 모든 카드의 위치는 게임이 종료될 때까지 변하지 않는다.    ***Parameters***  mIndexA : 선택할 카드의 위치 A (0 ≤ mIndexA ≤ 2N - 1)  mIndexB : 선택할 카드의 위치 B (0 ≤ mIndexB ≤ 2N - 1) |

**[예제]**

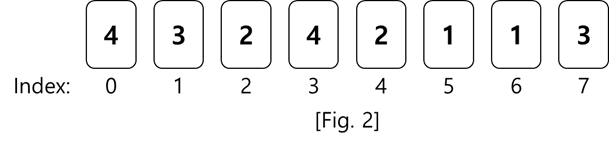
아래의 **[Table 2]**과 같이 입력되는 경우를 살펴보자.

|  |  |
| --- | --- |
| **Order** | **Function** |
| 1 | playGame(4) |

**[Table 2]**

나열된 카드에 4개의 서로 다른 페어가 존재한다.

총 8개의 카드가 존재하며 Sample input 에서 각 카드에 적힌 숫자를 확인하면 **[Fig. 2]**와 같다.



각 숫자가 적힌 카드의 위치를 차례대로 나열하면 아래와 같다.

숫자 1이 적힌 카드는 위치 5와 위치 6에 있다.

숫자 2가 적힌 카드는 위치 2과 위치 4에 있다.

숫자 3이 적힌 카드는 위치 1과 위치 7에 있다.

숫자 4가 적힌 카드는 위치 0과 위치 3에 있다.

따라서 **[Table 3]**에 표시된 순서대로 호출하는 경우 모든 페어를 찾을 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question** | | | **Answer** | **Found** |
| **Index A** | **Index B** | **Difference** |
| 5 | 6 | 0 | **TRUE** | **1** |
| 2 | 4 | 0 | **TRUE** | **2** |
| 1 | 7 | 0 | **TRUE** | **3** |
| 0 | 3 | 0 | **TRUE** | **4** |

**[Table 3]**

**[제약사항]**

1. 각 테스트 케이스 시작 시 playGame() 함수가 1회 호출된다.

**[입출력]**

입출력은 제공되는 Main 부분의 코드에서 처리하므로 User Code 부분의 코드에서는 별도로 입출력을 처리하지 않는다.

Sample input 에 대한 정답 출력 결과는 “#TC번호 결과” 의 포맷으로 보여지며 결과가 100 일 경우 정답, 0 일 경우 오답을 의미한다.