**[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

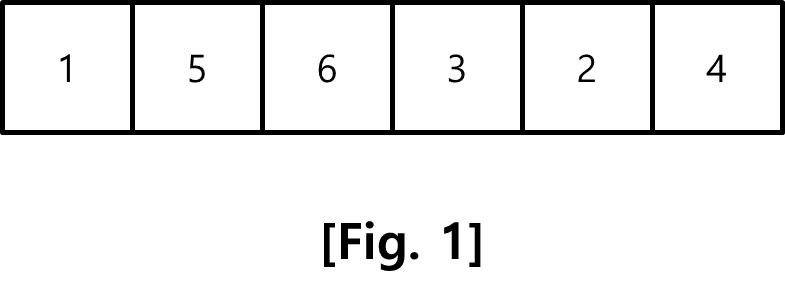
함께 제공되는 **Testcase 또한 학습을 위한 연습용**입니다.

|  |
| --- |
| **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 )  **ㅇ   B.**Main 부분의 코드를 main.cpp 또는 Solution.java 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]**

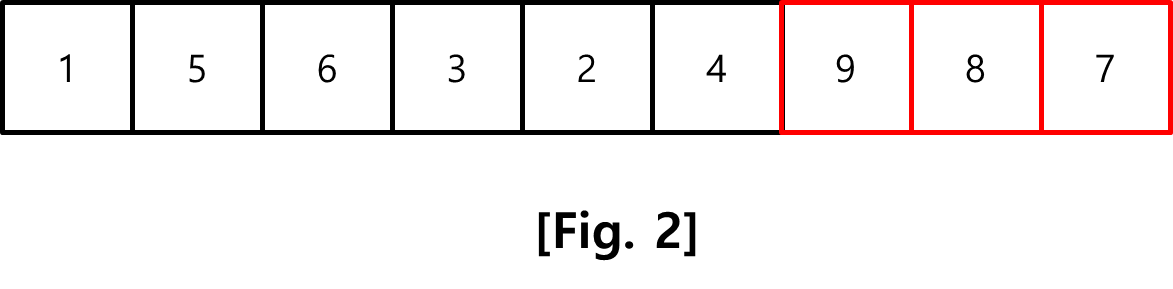
수가 순서대로 저장되어 있다.

**[Fig. 1]** 은 수 1, 5, 6, 3, 2, 4 왼쪽에서부터 저장되어 있는 상태를 나타낸다.



그런 상황에서 새로운 수들이 오른쪽 끝에 추가되어 저장 된다.

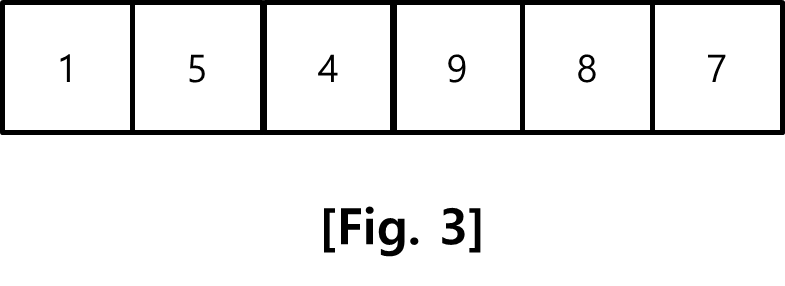
**[Fig. 2]** 는 **[Fig. 1]** 에서 수 9, 8, 7 가 추가된 후의 상태를 나타낸다.



또한 특정 구간의 수들이 삭제되기도 한다.

**[Fig. 3]** 는 **[Fig. 2]** 에서 3~5 번째 수가 삭제된 후의 상태를 나타낸다.

삭제 후 남아있는 수들은 그 순서를 유지한다.



이런 상황에서 오른쪽 첫번째부터 K 번째까지의 수 중 가장 큰 수와 가장 작은 수를 찾아 그 둘의 차이를 반환해야 한다.

**[Fig. 3]** 에서 K=5 일 경우, 당신이 반환해야 할 값은 9-4=5 이다.

아래 API 설명을 참조하여 각 함수를 구현하라.

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

|  |
| --- |
| **void init(int N, int mValue[])** |
| 각 테스트 케이스의 처음에 호출된다.    N 은 초기 상태에 저장해야 할 수의 개수이다.  mValue 에는 저장해야 할 수가 0번 인덱스부터 순서대로 저장되어 있다.    ***Parameters***     N: 저장해야 할 수의 개수 (1 ≤ N ≤ 30,000)     mValue: 저장해야 할 수 (0 ≤ mValue[i] ≤ 100,000,000) (단, 0 ≤ i ≤ N-1) |
| **void add(int M, int mValue[])** |
| 새로운 수 M 개를 기존에 저장되어 있는 수의 오른쪽 끝에 저장한다.  mValue 에는 저장해야 할 수가 0번 인덱스부터 순서대로 저장되어 있다.    ***Parameters***     M: 저장해야 할 수의 개수 (1 ≤ M ≤ 100)     mValue: 저장해야 할 수 (0 ≤ mValue[i] ≤ 100,000,000) (단, 0 ≤ i ≤ M-1) |
| **void erase(int mFrom, int mTo)** |
| 제일 앞에 있는 수를 1번째 수라고 정의했을 때, mFrom 번째 수부터 mTo 번째 수까지 삭제한다.  삭제 후 남아 있는 수들은 그 순서를 유지한다.  **[Fig. 3]** 을 참고하라    "mTo - mFrom" 은 0 이상 100 이하임이 보장된다.  mFrom ~ mTo 번째 수가 존재함이 보장된다.    ***Parameters***     mFrom : 삭제해야 할 수의 시작 위치     mTo : 삭제해야 할 수의 마지막 위치 |
| **int find(int K)** |
| 오른쪽 첫번째 수부터 K 번째까지의 수 중 가장 큰 수와 가장 작은 수의 차이를 반환한다.  K는 현재 저장되어 있는 수의 개수 이하임이 보장된다.    ***Parameter***    K: 결과값을 찾아야 할 수의 개수 (1 ≤ K)    ***Return***    K 개의 수 중 가장 큰 수와 작은 수의 차이 |

**[예제]**

다음은 첫번째 테스트케이스 호출되는 함수를 순서대로 표기하였다.

먼저 init(N=20, mValue={9, 12, 7, 8, 5, 9, 10, 8, 12, 13, 11, 15, 1, 4, 9, 12, 7, 8, 4, 14})이 호출되었다.

그 후 2번~18번째 함수는 다음과 같이 호출된다.

Status 는 현재 저장되어 있는 수를 순서대로 나타내며, 편의 상 1줄에 10개씩 표기하였다.

**검은색으로 굵게 표시된 수**는 find(K) 에서 해당되는 오른쪽 첫번째부터 K번째까지의 수를 의미한다.

**빨간색으로 굵게 표시된 수**는 Erase() 에 의해 삭제되는 수들을 의미한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Order** | **Function** | **Status** | **max-min** | **Return** |
| 2 | find(6) | 9 12 7 8 5 9 10 8 12 13  11 15 1 4 **9 12 7 8 4 14** | **14-4** | **10** |
| 3 | find(9) | 9 12 7 8 5 9 10 8 12 13  11 **15 1 4 9 12 7 8 4 14** | **15-1** | **14** |
| 4 | add(3,  {100, 8, 99}) | 9 12 7 8 5 9 10 8 12 13  11 15 1 4 9 12 7 8 4 14  100 8 99 |  |  |
| 5 | find(5) | 9 12 7 8 5 9 10 8 12 13  11 15 1 4 9 12 7 8 **4 14**  **100 8 99** | **100-4** | **96** |
| 6 | find(2) | 9 12 7 8 5 9 10 8 12 13  11 15 1 4 9 12 7 8 4 14  100 **8 99** | **99-8** | **91** |
| 7 | erase(19, 21) | 9 12 7 8 5 9 10 8 12 13  11 15 1 4 9 12 7 8 **4 14**  **100** 8 99 |  |  |
| 8 | find(10) | 9 12 7 8 5 9 10 8 12 13  **11 15 1 4 9 12 7 8 8 99** | **99-1** | **98** |
| 9 | erase(9, 12) | 9 12 7 8 5 9 10 8 **12 13**  **11 15** 1 4 9 12 7 8 8 99 |  |  |
| 10 | find(11) | 9 12 7 8 5 **9 10 8 1 4**  **9 12 7 8 8 99** | **99-1** | **98** |
| 11 | add(5, {101, 102, 103, 98, 97}) | 9 12 7 8 5 9 10 8 1 4  9 12 7 8 8 99 101 102 103 98  97 |  |  |
| 12 | find(7) | 9 12 7 8 5 9 10 8 1 4  9 12 7 8 **8 99 101 102 103 98**  **97** | **103-8** | **95** |
| 13 | find(2) | 9 12 7 8 5 9 10 8 1 4  9 12 7 8 8 99 101 102 103**98**  **97** | **98-97** | **1** |
| 14 | erase(17, 19) | 9 12 7 8 5 9 10 8 1 4  9 12 7 8 8 99 **101 102 103** 98  97 |  |  |
| 15 | find(5) | 9 12 7 8 5 9 10 8 1 4  9 12 7 **8 8 99 98 97** | **99-8** | **91** |
| 16 | erase(16, 18) | 9 12 7 8 5 9 10 8 1 4  9 12 7 8 8 **99 98 97** |  |  |
| 17 | find(3) | 9 12 7 8 5 9 10 8 1 4  9 12 **7 8 8** | **8-7** | **1** |
| 18 | find(6) | 9 12 7 8 5 9 10 8 1 **4**  **9 12 7 8 8** | **12-4** | **8** |

**[제약사항]**

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.

2. 각 테스트케이스에서 init() 함수와 add() 함수에서 전달된 수의 개수의 총합은 200,000 개 이하이다.

3. 각 테스트 케이스에서 add() 의 호출 횟수는 최대 2,000 회 이다.

4. 각 테스트 케이스에서 erase() 의 호출 횟수는 최대 500 회 이다.

5. 각 테스트 케이스에서 find() 의 호출 횟수는 최대 30,000 회 이다.

**[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