**[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]**

N개의 사물함을 배정하는 시스템이 있다.

사물함은 동일 간격으로 일렬로 존재하며, 1번부터 N번까지 차례로 번호가 주어진다.

사용자가 입실하는 경우, 아래 규칙에 따라 사물함을 배정한다.

**- 모든 사물함이 비어 있는 경우, 1번 사물함을 배정한다.**

**- 그렇지 않은 경우, 연속된 빈 사물함이 가장 큰 구간을 먼저 선택한다. 가장 큰 구간이 2개 이상일 경우, 사물함 번호가 작은 구간을 선택한다.  
  선택된 구간 내에서, 사용 중인 사물함과 거리가 가장 먼 곳의 사물함을 배정한다. 조건에 해당하는 사물함이 2개 이상일 경우, 번호가 작은 사물함을 배정한다.**

사용자가 퇴실하는 경우, 해당 사용자에게 배정되었던 사물함을 해제한다.

위와 같이 동작하는 사물함 배정 시스템을 구현하여 보자.

아래 API 설명을 참조하여 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 init(int N)** |
| 각 테스트 케이스의 처음에 호출된다.  N개의 사물함이 주어지며, 모든 사물함은 빈 상태이다.    ***Parameters***  N: 사물함의 개수 ( 8 ≤ N ≤ 100,000,000 ) |
| **int arrive(int mId)** |
| mId 사용자가 입실한다.  **mId가 사물함을 이용 중인 사용자 ID로 주어지는 경우는 없다.**  **빈 사물함이 없는 상황에서, 이 함수가 호출되는 경우는 없다.**  배정 규칙에 따라 사물함을 배정한다.  배정된 사물함의 번호를 반환한다.    ***Parameters***  mId: 사용자 ID ( 1 ≤ mId ≤ 1,000,000,000 )    ***Returns***  배정된 사물함의 번호를 반환한다. |
| **int leave(int mId)** |
| mId 사용자가 퇴실한다.  **mId는 사물함을 이용 중인 사용자 ID로만 주어진다.**  mId 사용자에게 배정되었던 사물함을 해제한다. 즉, 해당 사물함은 빈 사물함이 된다.  해제 후에 빈 사물함의 개수를 반환한다.    ***Parameters***  mId: 사용자 ID ( 1 ≤ mId ≤ 1,000,000,000 )    ***Returns***  빈 사물함의 개수를 반환한다. |

**[예제]**

아래의 **[Table 1]**과 같이 요청이 되는 경우를 살펴보자.

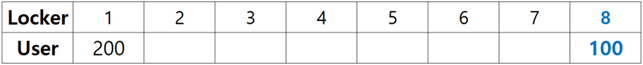
|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Function** | **return** | **Figure** |
| 1 | init(8) |  |  |
| 2 | arrive(200) | 1 |  |
| 3 | arrive(100) | 8 | Fig. 1 |
| 4 | arrive(700) | 4 | Fig. 2 |
| 5 | arrive(600) | 6 | Fig. 3 |
| 6 | leave(200) | 5 | Fig. 4 |
| 7 | arrive(300) | 1 | Fig. 5 |
| 8 | arrive(800) | 2 | Fig. 6 |
| 9 | arrive(200) | 3 | Fig. 7 |
| 10 | leave(600) | 3 |  |
| 11 | leave(100) | 4 | Fig. 8 |
| 12 | arrive(400) | 8 | Fig. 9 |

**[Table 1]**

**(순서 1)** 8개의 사물함이 주어진다. 초기에는 사물함이 모두 비어 있다.

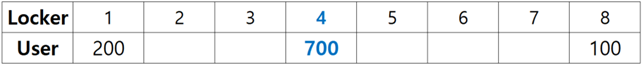
**(순서 2)** 200 사용자가 입실한다. 모든 사물함이 비어 있기 때문에 1번 사물함을 배정한다.

**(순서 3)** 100 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 2번 ~ 8번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 8번 사물함이므로 8을 반환한다. **[Fig. 1]**은 함수 호출의 결과를 나타낸 그림이다.



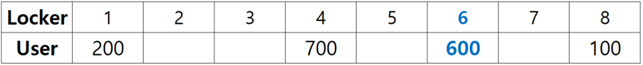
**[Fig. 1]**

**(순서 4)** 700 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 2번 ~ 7번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 4번, 5번 사물함이다. 이 중에서 번호가 작은 4번 사물함을 배정하고 4를 반환한다.  
**[Fig. 2]**는 함수 호출의 결과를 나타낸 그림이다.



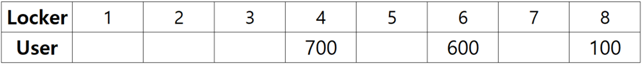
**[Fig. 2]**

**(순서 5)** 600 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 5번 ~ 7번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 6번 사물함이므로 6을 반환한다. **[Fig. 3]**은 함수 호출의 결과를 나타낸 그림이다.



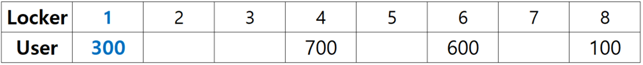
**[Fig. 3]**

**(순서 6)** 200 사용자가 퇴실한다. 1번 사물함은 빈 사물함이 된다. 빈 사물함의 총 개수로 5를 반환한다.  
**[Fig. 4]**는 함수 호출의 결과를 나타낸 그림이다.



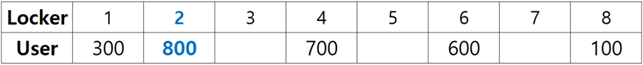
**[Fig. 4]**

**(순서 7)** 300 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 1번 ~ 3번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 1번 사물함이므로 1을 반환한다. **[Fig. 5]**는 함수 호출의 결과를 나타낸 그림이다.



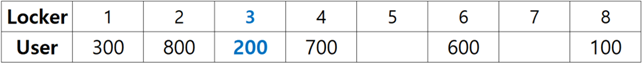
**[Fig. 5]**

**(순서 8)** 800 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 2번 ~ 3번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 2번, 3번 사물함이다. 이 중에서 번호가 작은 2번 사물함을 배정하고 2를 반환한다.  
**[Fig. 6]**은 함수 호출의 결과를 나타낸 그림이다.



**[Fig. 6]**

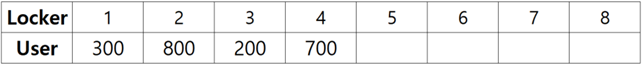
**(순서 9)** 200 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 3번 ~ 3번 구간, 5번 ~ 5번 구간, 7번 ~ 7번 구간이다.  
이 중에서 사물함 번호가 작은 3번 ~ 3번 구간을 선택한다. 사용 중인 사물함과 가장 먼 곳은 3번 사물함이므로 3을 반환한다.  
**[Fig. 7]**은 함수 호출의 결과를 나타낸 그림이다.



**[Fig. 7]**

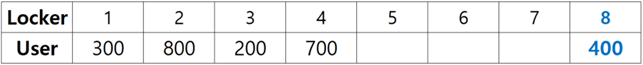
**(순서 10)** 600 사용자가 퇴실한다. 6번 사물함은 빈 사물함이 된다. 빈 사물함의 총 개수로 3을 반환한다.

**(순서 11)** 100 사용자가 퇴실한다. 8번 사물함은 빈 사물함이 된다. 빈 사물함의 총 개수로 4를 반환한다.  
**[Fig. 8]**은 함수 호출의 결과를 나타낸 그림이다.



**[Fig. 8]**

**(순서 12)** 400 사용자가 입실한다. 연속된 빈 사물함이 가장 큰 구간은 5번 ~ 8번 구간이다.  
사용 중인 사물함과 가장 먼 곳은 8번 사물함이므로 8을 반환한다. **[Fig. 9]**는 함수 호출의 결과를 나타낸 그림이다.



**[Fig. 9]**

**[제약사항]**

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.

2. 각 테스트 케이스에서 arrive() 함수의 호출 횟수는 20,000 이하이다.

3. 각 테스트 케이스에서 모든 함수의 호출 횟수 총합은 35,000 이하이다.

**[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.