**[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]**

PC가 접속해서 데이터를 다운로드 할 수 있는 다운로드 서비스를 시뮬레이션 하려고 한다.

다운로드 서비스는 다음 3가지 요소를 가지는 트리 구조이다.

- 서버 : 트리의 루트(root)에 해당하며, 전송 속도 mCapa 로 데이터를 제공한다.

- PC : 트리의 잎(leaf)의 일부가 해당하며, 다운로드 서비스에 접속해 원하는 크기만큼 데이터를 다운로드 하고 다운로드 완료 후 연결을 해제한다.

- 허브 : 서버 또는 부모 허브에서 받은 데이터를 자식 허브 또는 PC에 전달하는 역할을 한다.

데이터 다운로드는 PC 만 요청하고, 다운로드 완료된 PC 는 자동으로 접속 해제한다.

전송 속도는 데이터를 다운로드 하는 속도를 의미하며, PC 는 1 시간에 주어진 전송 속도만큼 데이터를 다운로드한다. 예를 들어, 전송 속도가 4 이면 PC 는 1 시간에 크기가 4 인 데이터를 다운로드한다.

서버 또는 허브는 각각 자신과 연결된 자식 허브 또는 PC 중에서 다운로드가 필요한 자식 허브 또는 PC 에게 균등하게 전송 속도 분배하여 데이터를 전달한다.

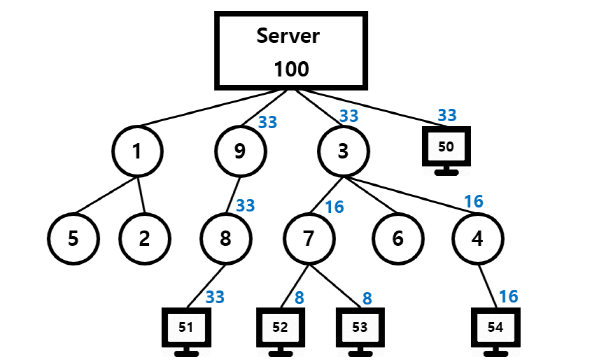
**단, 서버 또는 부모 허브에서 받은 전송 속도를 균등하게 분배하되, 소수점 이하는 버림한다.**

**다운로드가 필요한 허브는 각각의 PC 가 서버에서 데이터를 다운로드 하는데 필요한 허브이다.**

**즉, 다운로드가 필요한 허브는 자신 또는 자신의 후손 허브들 중에서 PC 와 연결된 허브가 있는 경우이다.**

예를 들어, [Fig. 1]과 같이 서버, 허브, PC 가 연결되어 있는 경우를 생각해 보자.

서버는 100의 전송 속도를 제공하고, 1, 2, 3, 4, 5, 6, 7, 8, 9 번은 허브이고, 50, 51, 52, 53, 54 번은 PC 이다.



**[Fig. 1]**

여기에서 다운로드가 필요한 허브는 9, 3, 8, 7, 4 번 허브이다.

54번 PC 가 마지막에 추가된다고 하면, 추가된 후 다운로드 서비스의 전송 속도는 [Fig. 1] 과 같이 변경된다.

( 허브, PC 안의 숫자는 각각의 ID 를, 위의 파란색 숫자는 각각의 전송 속도이다. )

[Fig. 1] 과 같이 서버 또는 부모 허브는 데이터 다운로드가 필요한 자식 허브들과 PC 에게만 전송 속도를 균등하게 분배한다. 소수점 이하는 버림한다.

서버는 자신과 연결되고 다운로드가 필요한 허브와 PC 에게 균등하게 전송 속도를 분배하기 때문에, 9, 3번 허브와 50번 PC 에 각각 33 의 전송 속도를 분배한다.

3번 허브는 서버에서 받은 33 전송 속도를 다운로드가 필요한 7, 4번 2개의 자식 허브 에게 16 의 전송 속도를 균등하게 분배한다.

만약, 부모 허브에서 받은 전송 속도가 2 이고 다운로드가 필요한 자식 허브가 3 개 일 경우, 자식 허브들은 0 의 전송 속도를 가진다.

서버, 허브, PC 는 각각 ID를 가진다. **서버, 허브, PC 중에 같은 ID를 가지는 경우는 없다.**

서버의 ID는 언제나 0 이다.

매 시각, 허브는 추가 및 삭제 될 수 있으며 데이터를 다운로드 하기 위해 PC가 추가 될 수 있다.

크기가 15 인 데이터를 원하는 PC 가 10 시각에 추가되면, 다운로드 서비스의 전송 속도가 변경 된다. 이로 인해 PC 가 4 의 전송 속도를 제공 받을 경우, 14 시각에 다운로드를 완료할 수 있다.

다운로드 완료 후 PC 는 접속을 해제하고 다운로드 서비스는 14 시각에 전송 속도를 다시 변경한다.

즉, **PC 가 추가 되거나 다운로드 완료하여 자동으로 접속 해제될 때, 즉시 다운로드 서비스의 전송 속도는 변경된다.**

이러한 다운로드 서비스를 시뮬레이션하는 프로그램을 작성하자.

아래 API 설명을 참조하여 각 함수를 구현하라.

※ 아래 함수 signature는 C/C++에 대한 것으로 Java에 대해서는 제공되는 Solution.java와 UserSolution.java를 참고하라.

아래는 User Code 부분에 작성해야 하는 API 의 설명이다.

|  |
| --- |
| **void init(int mCapa)** |
| 각 테스트 케이스의 처음에 호출된다.    다운로드 서비스에서 트리의 루트에 해당하는 서버만 있다.  서버는 mCapa 의 전송 속도를 가진다.    서버의 ID 는 0 이다.  현재 시각은 0 이다. ( mTime = 0)      ***Parameters***    mCapa : 서버가 제공하는 전송 속도 ( 100 ≤ mCapa ≤ 50,000 ) |
| **void addHub(int mTime, int mParentID, int mID)** |
| mTime 시각에 ID가 mParentID 인 서버 또는 허브에 mID 자식 허브를 추가한다.    mTime 시각까지 다운로드 실행 후, mID 허브를 추가한다.    mParentID 는 삭제된 허브의 ID 가 아니고, PC 의 ID 가 아님을 보장한다.  mID 허브는 이전에 해당 함수에서 호출된 적이 없다.      ***Parameters***    mTime    : mID 허브를 추가하는 시각 ( 1 ≤ mTime ≤ 50,000 )    mParentID : 서버 또는 부모 허브의 ID ( 0 ≤ mParentID ≤ 50,000 )    mID      : 추가하는 자식 허브의 ID ( 1 ≤ mID ≤ 50,000 ) |
| **int removeHub(int mTime, int mID)** |
| mTime 시각에 mID 허브를 삭제한다.  허브 삭제에 성공한 경우 1 을 반환하고, 실패하면 0 을 반환한다.    mTime 시각까지 다운로드 실행 후, 허브 삭제 여부를 확인한다.  (mTime 시각에 다운로드가 완료된 PC가 있는 경우, PC를 접속 해제한다. 그 이후 mID 허브 삭제 여부를 확인한다.)    mID 허브 또는 허브의 자손이 PC 와 연결되어 있을 경우 삭제에 실패한다.    mID 허브는 addHub() 에서 추가된 허브이다.  mID 허브는 삭제된 허브가 아니고, PC 가 아님이 보장된다.    mID 허브 삭제 시 자손 허브들 역시 모두 삭제된다.      ***Parameters***    mTime   : mID 허브를 삭제하는 시각 ( 1 ≤ mTime ≤ 50,000 )    mID     : 삭제하려고 하는 허브의 ID ( 1 ≤ mID ≤ 50,000 )    ***Returns***    허브의 삭제 성공 여부 |
| **void requestDL(int mTime, int mParentID, int mpcID, int mSize)** |
| mTime 시각에 ID 가 mParentID 인 서버 또는 허브에 mpcID 의 PC 를 추가한다.  mpcID 의 PC 는 mSize 크기만큼 데이터 다운로드를 시작한다.    mTime 시각까지 데이터 다운로드 실행 후, mpcID 의 PC 가 추가된다.    mTime 시각에 데이터 다운로드가 완료된 PC 는 자동 접속 해제되며, 즉시 다운로드 서비스의 전송 속도를 변경한다. mpcID 의 PC 가 추가 후, 다운로드 서비스는 다시 전송 속도를 변경한다.    mParentID 는 삭제된 허브의 ID 가 아니고, PC 의 ID 가 아님이 보장된다.  mpcID 의 PC 는 이전에 호출된 적이 없다.      ***Parameters***    mTime    : PC 를 추가하는 시각 ( 1 ≤ mTime ≤ 50,000 )    mParentID : 서버 또는 부모 허브의 ID ( 0 ≤ mParentID ≤ 50,000 )    mpcID    : 데이터를 다운로드 하는 PC의 ID ( 1 ≤ mpcID ≤ 50,000 )    mSize    : PC 의 데이터 다운로드 크기 (1 ≤ mSize ≤ 1,000,000 ) |
| **Result checkPC(int mTime, int mpcID)** |
| mTime 시각에 mpcID 의 PC 상태를 확인한다.    mTime 시각까지 데이터 다운로드 실행 후, mpcID 의 PC 상태를 확인한다.    데이터를 다운로드 하고 있는 경우 Result 구조 내의 finish = 0, param = 남아 있는 다운로드 크기  데이터 다운로드가 완료된 경우 Result 구조 내의 finish = 1, param = 완료된 시각을 반환한다.    mpcID 의 PC 는 requestDL() 에서 호출된 PC 이다.      ***Parameters***    mTime  : PC 의 상태를 확인하는 시각 ( 1 ≤ mTime ≤ 50,000 )    mpcID  : 상태를 확인하는 PC의 ID ( 1 ≤ mpcID ≤ 50,000 )    ***Returns***  데이터 다운로드 하고 있는 경우, finish = 0, param = 남아 있는 다운로드 크기    데이터 다운로드 완료한 경우, finish = 1, param = 완료된 시각 |

**[예제]**

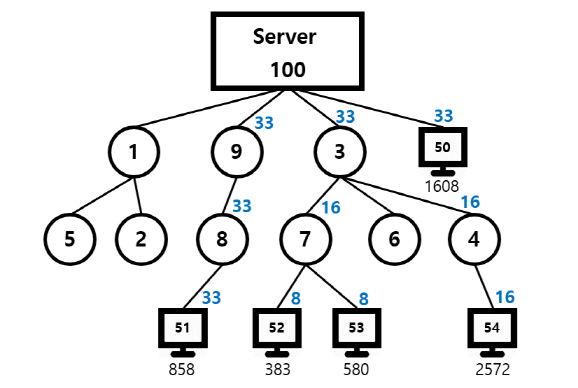
아래에는 테스트케이스 1번을 처음부터 순서대로 보여준다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Function** | **Note** | **Figure** |
| 1 | init(100) |  |  |
| 2 | addHub(1, 0, 1) |  |  |
| 3 | addHub(2, 0, 9) |  |  |
| 4 | addHub(3, 0, 3) |  |  |
| 5 | requestDL(5, 0, 50, 3120) |  |  |
| 6 | addHub(6, 1, 5) |  |  |
| 7 | addHub(7, 1, 2) |  |  |
| 8 | addHub(8, 9, 8) |  |  |
| 9 | addHub(9, 3, 7) |  |  |
| 10 | addHub(10, 3, 6) |  |  |
| 11 | addHub(11, 3, 4) |  |  |
| 12 | requestDL(15, 8, 51, 1370) |  |  |
| 13 | requestDL(16, 7, 52, 675) |  |  |
| 14 | requestDL(20, 7, 53, 740) |  |  |
| 15 | requestDL(30, 4, 54, 2572) |  | **[Fig. 2]** |

**[Table 1]**

Order 15. requestDL(30, 4, 54, 2572) 실행 시 데이터 다운로드 후 전송 속도는 [Fig. 2] 와 같다.

( Hub 와 PC 위의 파란색 숫자는 각각의 전송 속도이고, PC 밑의 숫자는 남아 있는 다운로드 크기이다. )



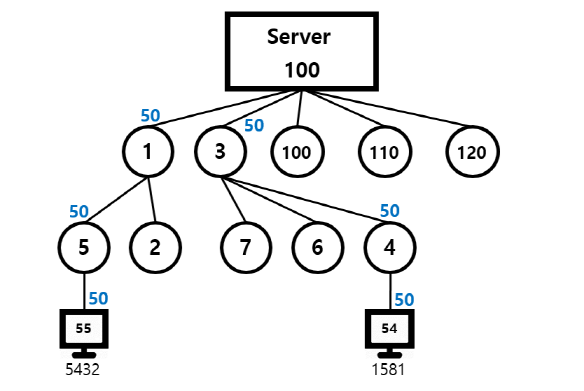
**[Fig. 2]**

[Table 2] 는 [Table 1] 이후의 함수 호출이다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Function** | **Note** | **Figure** |
| 16 | removeHub(50, 9) | return 0 |  |
| 17 | removeHub(60, 9) | return 1 |  |
| 18 | checkPC(61, 50) | finish = 0, param = 500 |  |
| 19 | checkPC(63, 51) | finish = 1, param = 56 |  |
| 20 | checkPC(65, 52) | finish = 0, param = 67 |  |
| 21 | addHub(69, 0, 100) |  |  |
| 22 | addHub(70, 0, 110) |  |  |
| 23 | addHub(71, 0, 120) |  |  |
| 24 | requestDL(75, 5, 55, 5432) |  | **[Fig. 3]** |

**[Table 2]**

Order 24. requestDL(75, 5, 55, 5432) 실행 후 다운로드 서비스의 속도와 각 PC 의 남은 데이터는 [Fig. 3] 과 같다.



**[Fig. 3]**

[Table 3] 은 [Table 2] 이후의 함수 호출이다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Order** | **Function** | **Note** | **Figure** |
| 25 | checkPC(80, 50) | finish = 1, param = 71 |  |
| 26 | checkPC(85, 51) | finish = 1, param = 56 |  |
| 27 | checkPC(90, 52) | finish = 1, param = 71 |  |
| 28 | checkPC(91, 53) | finish = 1, param = 75 |  |
| 29 | checkPC(94, 54) | finish = 0, param = 631 |  |
| 30 | checkPC(99, 55) | finish = 0, param = 4232 |  |

**[Table 3]**

**[제약사항]**

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.

2. 각 테스트 케이스에서 서버, PC, 허브의 ID 값은 유일하다.

3. 각 테스트 케이스에서 mTime 은, 매 함수 호출시 마다 1 이상 증가하는 값이다.

4. **각 테스트 케이스에서 서버 또는 허브가 가질 수 있는 자식 허브(PC 포함)의 개수는 5개 이하이다.**

**( 이 때, 삭제된 자식 허브나 PC 는 포함하지 않는다. )**

5. 각 테스트 케이스에서 addHub() 함수의 호출 횟수는 10,000 이하이다.

6. 각 테스트 케이스에서 removeHub() 함수의 호출 횟수는 100 이하이다.

7. 각 테스트 케이스에서 requestDL(), checkPC() 함수의 호출 횟수는 각각 1,000 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.