SW Expert Academy 모의 테스트

박용성

제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

13453. [2022 동계 대학생 양성과정] 가족 구성원 찾기

문제 내용

시간 : 10개 테스트케이스를 합쳐서 C++의 경우 1초 / Java의 경우 2초

메모리 : 힙, 정적 메모리 합쳐서 256MB 이내, 스택 메모리 1MB 이내

- 지원언어 : C++ / JAVA

*** STL 라이브러리 사용 가능**

① **C 또는 C++로 답안을 작성하시는 응시자께서는 검정시스템에 제출 시, Language 에서 C++ 를 선택하신 후**

② **Main 과 User Code 부분으로 구성되어 있습니다.**

A. Main : 수정할 수 없는 코드이며, 채점 시 비 정상적인 답안 검출 등 평가를 위한 로직이 추가 될 수

B. User Code : 실제 응시자가 작성해야 하는 코드이며, 제출 시에는 표준 입출력 함수가 포함되어 있으면

③ **Local PC 에서 프로그래밍 시 유의 사항**

A. 2개의 파일을 생성하셔야 합니다. (main.cpp / solution.cpp 또는 Solution.java / UserSolution.java)

B. Main 부분의 코드를 main.cpp 또는 Solution.java 에 복사해서 사용하시기 바랍니다.

C. sample_input.txt 를 사용하시기 위해서는 Main 부분의 코드 내에

표준 입력을 파일로 전환하는 코드 (주석처리 되어 있음) 의 주석을 풀어서 사용하시면 됩니다.

D. User Code 부분의 코드를 작성하신 후 서버에 제출하실 때,

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]

가족 관계도를 작성하고 조회하는 프로그램을 개발할 계획이다.

이 프로그램은 다음과 같은 기능을 가지고 있다.

1. 구성원 추가 : 가족 관계도에 새로운 가족 구성원을 추가하고, 기존 가족 구성원과 관계를 설정한다.

2. 두 구성원 간의 친족 거리 조회 : 작성된 가족 관계도를 기반으로 두 구성원 간의 친족 거리를 조회한다.

3. 특정 친족 거리에 있는 구성원 수 세기 : 기준이 되는 구성원으로부터, 주어진 친족 거리에 있는 가족 구성원의 수를 센다.

새로운 구성원을 추가하는 경우, 기존 가족 구성원의 배우자, 부모, 또는 자식으로 추가할 수 있다.

세부 설명은 각각 다음과 같다.

1. **배우자** : 새 구성원이 기존 구성원과 서로 배우자가 된다.

제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

B. 추가적인 관계 : 배우자의 자식은 모두 새 구성원의 자식으로도 추가되며, 그 자식에게도 새 구성원이 부모로 추가된다.

2. 부모 : 새 구성원이 기존 구성원의 부모가 되고, 이 기존 구성원은 새 구성원의 자식이 된다.

A. 예외 : 기존 구성원에게 새 구성원과 성별이 같은 부모가 이미 있는 경우, 새 구성원은 기존 구성원의 부모가 될 수 없다.

B. 추가적인 관계 : 기존 구성원에게 새 구성원과 성별이 다른 부모가 있다면, 그 기존의 부모는 새 구성원과 서로 배우자가 된다.

또한, 이 경우에는 기존의 부모의 자식이 모두 새 구성원의 자식으로도 추가되며, 그 자식에게도 새 구성원이 부모로 추가된다.

3. 자식 : 새 구성원이 기존 구성원의 자식이 되고, 이 기존 구성원은 새 구성원의 부모가 된다.

A. 예외 : 이 경우는 예외가 존재하지 않는다.

B. 추가적인 관계 : 기존 구성원에게 배우자가 있다면, 새 구성원은 기존 구성원의 배우자의 자식으로도 추가되며,

기존 구성원의 배우자는 기존 구성원과 함께 새 구성원의 또 한 명의 부모가 된다.

가족 관계도에서 두 구성원의 **친족 거리**는 다음과 같다.

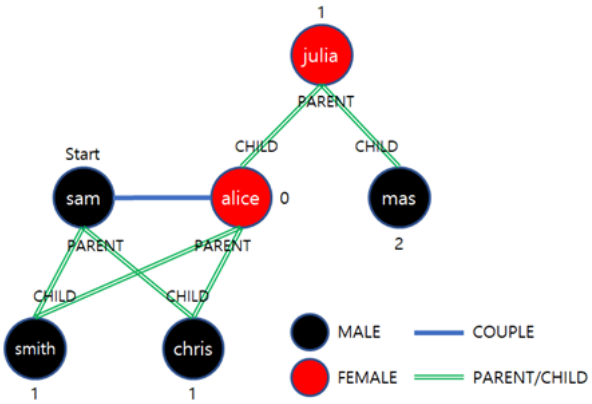
1. **배우자 사이의 친족 거리는 0** 이다.
2. **부모-자식 사이의 친족 거리는 1** 이다.
3. 이 외의 경우, **친족 거리는 두 구성원을 연결하는 경로의 친족 거리의 합의 최소값**이다. (아래 예시 참조)

[Fig. 1] 과 같이 가족 관계도가 구성된 경우를 살펴보자.

아래 그림들에서 MALE 은 남성, FEMALE 은 여성을 의미한다.

COUPLE 은 서로 배우자인 경우를 의미하고, PARENT 는 부모, CHILD 는 자식을 의미한다.

각 가족 구성원을 나타내는 원 근처에 표기된 숫자는 Start 가 표기된 "sam" 으로부터의 친족 거리를 의미한다.



[Fig. 1]

"sam" 과 "alice" 는 서로 배우자이므로 친족 거리가 0 이다.

"sam" 은 "chris" 의 부모이므로 친족 거리가 1 이다.

"sam" 과 "julia" 는 "sam" → "alice" → "julia" 로 연결되므로 친족 거리는 0 + 1 = 1 이다.

(여기서 "sam" → "chris" → "alice" → "julia" 로 연결될 수도 있지만, 이 경로의 친족 거리의 합 3 은 최소값이 아니므로 "sam" 과 "julia" 의 친족 거리가 아니다.)

"sam" 과 "mas" 는 "sam" → "alice" → "julia" → "mas" 로 연결되므로 친족 거리는 0 + 1 + 1 = 2 이다.

이와 같은 가족 관계도를 작성하고 조회하는 프로그램을 작성하라.

고하라.

아래는 User Code 부분에 작성해야 하는 API 의 설명이다.

void init(char initialMemberName[], int initialMemberSex)

가족 관계도를 초기화하는 함수로, 각 테스트 케이스의 처음에 1회 호출된다.

초기화 후, 가족 관계도에는 이름이 "initialMemberName" 이고 성별이 "initialMemberSex" 인 구성원 1명만 존재한다.

이름은 영어 소문자로 구성되어 있으며, '\0' 로 끝나고, '\0' 을 제외한 길이는 1 이상 19 이하이다.
성별은 남성의 경우 0 으로 주어지고, 여성의 경우 1 로 주어진다.

Parameters
initialMemberName : 가족 구성원의 이름
initialMemberSex : 가족 구성원의 성별

bool addMember(char newMemberName[], int newMemberSex, int relations

새 구성원을 가족 관계도에 추가하고 기존의 구성원과 관계를 설정한다.

newMemberName, newMemberSex 는 각각 새 구성원의 이름, 성별이고,
existingMemberName 은 새 구성원과 관계를 설정할 기존 구성원의 이름이다.
이름, 성별의 제약 사항은 init() 함수에서 설명한 것과 동일하다.

relationship 은 관계이고, 0, 1, 2 중 하나의 정수이며 의미는 다음과 같다.
0 : newMemberName 을 existingMemberName 의 **배우자**로 추가
1 : newMemberName 을 existingMemberName 의 **부모**로 추가
2 : newMemberName 을 existingMemberName 의 **자식**으로 추가

relationship 값에 따라 해당하는 본문 설명을 참고하여 **새 구성원을 추가하고 지정한 관계 및 추가적인 관계를 설정한 성공하는 경우 true 를 반환**하고, 만약, **예외에 해당한다면 실패하고 false 를 반환**한다.
실패하는 경우 가족 관계도에 변화가 없고, 새 구성원도 추가되지 않는다.

이 함수가 호출될 때 이름이 "newMemberName" 인 구성원은 가족 관계도에 **없음**이 보장된다.
이 함수가 호출될 때 이름이 "existingMemberName" 인 구성원은 가족 관계도에 **있음**이 보장된다.

Parameters
newMemberName : 새 구성원의 이름
newMemberSex : 새 구성원의 성별
relationship : 새 구성원과 기존 구성원 과의 관계
existingMemberName : 기존 구성원의 이름
Returns
새 구성원의 추가 및 관계 설정에 성공하면 true, 실패하면 false

int getDistance(char nameA[], char nameB[])

이름이 "nameA" 인 구성원과 이름이 "nameB" 인 구성원 간의 친족 거리를 반환한다.

이름의 제약 사항은 init() 함수에서 설명한 것과 동일하다.
이 함수에서 주어지는 두 구성원은 가족 관계도에 **있음**이 보장되고, **서로 다른 구성원**임이 보장된다.

Parameters
nameA, nameB : 친족 거리를 확인하고자 하는 두 구성원의 이름
Returns
주어진 두 구성원 사이의 친족 거리

int countMember(char name[], int dist)

이름이 "name" 인 구성원과 친족 거리가 "dist" 인 가족 구성원을 찾고, 그 수를 반환한다.
반환하는 수에 본인은 포함되지 않는다.

이름의 제약 사항은 init() 함수에서 설명한 것과 동일하다.
이 함수에서 주어지는 구성원은 가족 관계도에 **있음**이 보장된다.

Parameters
name : 기준이 되는 구성원의 이름
dist : 반환할 대상을 찾을 친족 거리 (dist ≥ 0)
Returns
주어진 구성원과 dist 만큼의 친족 거리를 갖는 구성원의 수

[예제]

Order	Function	Return	Figure
1	init("sam", 0)		
2	addMember("smith", 0, 2, "sam")	true	

제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

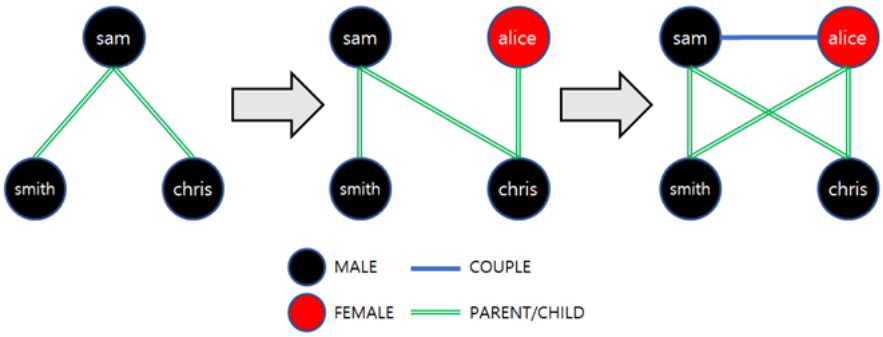
제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

6	addMember("mas", 0, 2, "julia")	true	
7	addMember("tony", 0, 0, "alice")	false	[Fig. 3]
8	addMember("mak", 0, 1, "smith")	false	[Fig. 3]
9	addMember("rainy", 1, 1, "smith")	false	[Fig. 3]
10	addMember("dandy", 0, 0, "julia")	true	[Fig. 4]
11	getDistance("sam", "chris")	1	[Fig. 4]
12	getDistance("sam", "dandy")	1	[Fig. 4]
13	getDistance("smith", "alice")	1	[Fig. 4]
14	getDistance("mas", "sam")	2	[Fig. 4]
15	getDistance("alice", "sam")	0	[Fig. 4]
16	addMember("joon", 0, 2, "dandy")	true	
17	addMember("frank", 0, 0, "joon")	false	
18	addMember("anne", 1, 0, "joon")	true	
19	addMember("david", 0, 2, "anne")	true	
20	addMember("munk", 0, 1, "david")	false	
21	addMember("john", 0, 1, "alice")	false	
22	getDistance("david", "smith")	4	
23	countMember("sam", 0)	1	[Fig. 5]
24	countMember("sam", 1)	4	[Fig. 5]
25	countMember("sam", 2)	3	[Fig. 5]
26	countMember("sam", 3)	1	[Fig. 5]
27	countMember("sam", 4)	0	[Fig. 5]

아래 그림 [Fig. 2] 에서 addMember() 함수를 통한 구성원 추가 및 관계 설정 과정을 살펴보자.



[Fig. 2]

위 그림은 “sam” 이 “smith” 와 “chris” 의 부모인 상황에서 addMember("alice", 1, 1, "chris") 함수가 호출된 상황이다.

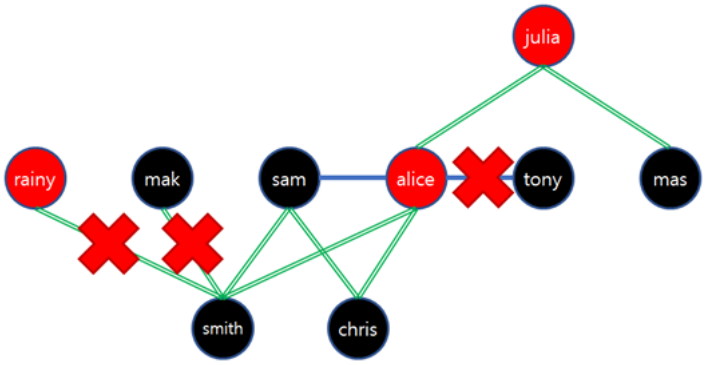
이 함수의 의미는 성별이 여성인 “alice” 를 “chris” 의 부모로 추가하라는 뜻이다.

성별이 남성인 “sam” 이 이미 “chris” 의 부모이므로, “alice” 는 “sam” 과 서로 배우자이다.

“chris” 외에 “sam” 의 자식인 “smith” 는 “sam” 의 배우자인 “alice” 의 자식이 됨을 알 수 있다.

즉, addMember("alice", 1, 1, "chris") 의 호출 결과는 [Fig. 2] 의 가장 우측 그림과 같이 됨을 알 수 있다.

아래 그림 [Fig. 3] 은 Order 7~9 의 구성원을 추가할 수 없는 상황을 보여준다.



[Fig. 3]

“alice” 는 배우자가 이미 있기 때문에 “tony” 를 “alice” 의 배우자로 추가할 수 없다.

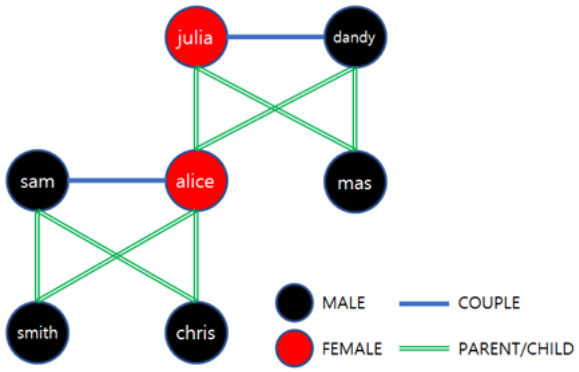
제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

따라서, Order 7~9 의 addMember() 함수 호출 결과로 false 를 반환하고, 구성원도 추가되지 않는다.

[Fig. 4] 는 "dandy" 까지 추가된 가족 관계도 그림이다.



[Fig. 4]

이 상황에서 두 구성원 사이의 친족 거리를 측정하는 getDistance() 함수의 호출 결과를 살펴보자.

getDistance("sam", "chris") : "sam" → "chris" : 1

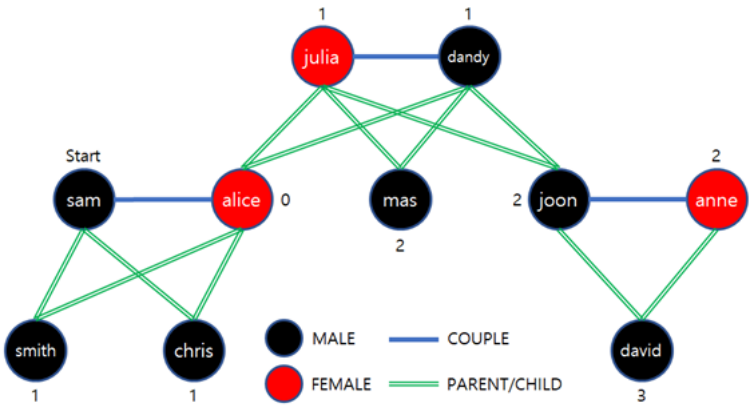
getDistance("sam", "dandy") : "sam" → "alice" → "dandy" : 0 + 1 = 1

getDistance("smith", "alice") : "smith" → "alice" : 1

getDistance("mas", "sam") : "mas" → ("dandy" 또는 "julia") → "alice" → "sam" : 1 + 1 + 0 = 2

getDistance("alice", "sam") : "alice" → "sam" : 0

마지막으로 가족 관계도 구성이 모두 완료된 후의 그림 [Fig. 5] 를 살펴보자.



[Fig. 5]

[Fig. 5] 에는 각 구성원의 "sam" 으로부터의 친족 거리를 가족 관계도에 숫자로 표시하였다.

"sam" 을 기준으로 countMember() 함수를 호출하면, 주어진 친족 거리에 해당하는 구성원의 수를 세어 반환한다.

예를 들어, countMember("sam", 2) 가 호출되면 "sam" 으로부터 친족 거리 2 에 해당하는 "mas", "joon", "anne" 이 그 대상이 되고, 3 을 반환한다.

[제약사항]

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.

2. 이름은 영어 소문자로 구성되어 있으며, '\0' 로 끝나고, '\0' 을 제외한 길이는 1 이상 19 이하이다.

3. 각 테스트 케이스에서 같은 이름의 구성원은 둘 이상 존재하지 않는다.

4. 각 테스트 케이스에서 가족 구성원의 수는 최대 200 이다.

5. 각 테스트 케이스에서 모든 함수의 총 호출 횟수는 최대 100,000 이다.

SW Expert Academy 모의 테스트

박용성

제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

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

입력

```
5 100
27
0 sam 0
1 smith 0 2 sam 1
1 chris 0 2 sam 1
1 alice 1 1 chris 1
1 julia 1 1 alice 1
1 mas 0 2 julia 1
....

// 첨부파일 sample_input.txt 참조
```

sample_input.txt

출력

```
#1 100
#2 100
#3 100
#4 100
#5 100
```

sample_output.txt

문제 풀이

Language C++14 (gcc-10.3)

Main

```
#ifndef _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#endif

#include <stdio.h>

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
#define MALE   0
#define FEMALE 1

#define INIT         0
#define ADDMEMBER    1
#define GETDISTANCE  2
#define COUNTMEMBER  3

#define COUPLE 0
```

User Code

```
//   The below commented functions are for your reference. If you want
//   to use it, uncomment these functions.
/*
int mstrcmp(const char a[], const char b[])
{
    int i;
    for (i = 0; a[i] != '₩0'; ++i) if (a[i] != b[i]) return a[i] - b[i];
    return a[i] - b[i];
}

void mstrcpy(char dest[], const char src[])
{
```

SW Expert Academy 모의 테스트

박용성

16 }

소스 코드 초기화

제출결과

제출횟수 0 / 10

제출회차	제출시간	결과

TEST

Input 을 입력하고 Run을 선택하면 Output 결과를 확인할 수 있습니다. Input 값을 넣지 않으면 기본 Input값이 적용되어 실행됩니다. Test는 채점을 하는 것이 아니며 정답 여부를 알려주지 않습니다.

Input

Input값을 입력해 주세요.

Output

Run Clear

임시저장 컴파일 제출