**[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 |

|  |
| --- |
| **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]**

수식들이 주어지면,  
이전에 주어졌던 수식들 중 동일한 수식들을 찾거나 수식을 계산하는 프로그램을 작성하려고 한다.

주어지는 수식은 아래와 같은 특성을 가지고 있다.  
- 수식은 길이가 200 이하인 문자열로 나타낸다.  
- 수식마다 유일한 ID가 주어지며, 이 값을 이용하여 수식들을 구분할 수 있다.  
- 수식을 나타내는 문자열이 같아도 ID가 다르면 다른 수식으로 간주한다.  
- 수식은 변수, 수, 연산자, 괄호로 구성되며 수식 내에 빈 공간은 존재하지 않는다.  
- 하나의 수식 내에는 **최대 3종류**의 변수들이 포함될 수 있다.  
- 변수의 이름은 영문 대문자인 ‘A’, ‘B’, ‘C’ 중 하나로 주어진다.  
- 수식 내의 수는 1부터 5까지의 한자리 수로만 주어진다.  
- 수식 내에는 연산자로 +(더하기), -(빼기), \*(곱하기)만 주어진다.  
- **하나의 수식 내에 \*(곱하기) 기호는 최대 한 개만 존재함을 보장한다.**  
- 수식 계산 시 일반적인 연산자 우선순위를 사용하여 계산한다.  
- 괄호의 개수는 제한이 없으며, 하나의 수나 변수를 둘러싸는 괄호들도 존재할 수 있다.  
- 수식의 맨 앞 또는 열린 괄호 "(" 바로 오른쪽에 연산자가 올 수 없다.  
  예를 들어 -4\*2, -A+3, (-2+1) 등은 허용되지 않는다.

주어지는 수식의 예는 아래와 같다.

(((3)+5)+2)  
(A\*B-C)+5  
A+B+C+(A+B+C+5)\*(A+B+C+5)+A+B+C+5  
(A+2)\*(B-2)  
(A-A)\*(B-B)

아래와 같은 수식은 주어지지 않는다.

((3+1)+0 : 괄호 쌍이 일치해야 하며 0은 주어지지 않는다.  
(A+B+C+D)\*(15) : 변수 명은 A, B, C 3가지만 존재하며, 수는 5이하만 주어질 수 있다.  
-5\*2-A : ‘-‘ 연산 기호가 수식의 맨 앞에 나올 수 없다.  
A\*A\*A : 곱하기 연산자는 최대 한 개만 포함될 수 있다

주어지는 수식에 포함된 변수들 중, 대입된 값에 따라 수식의 계산 결과 값을 변경시키는 변수를  
“**진짜 변수**”라고 하자.

예를 들어 수식이 “A+5”인 경우  
A에 3을 대입하면 결과는 8이 되지만 A에 5를 대입하면 계산 결과가 10이 되어  
대입된 값이 다르면 계산 결과가 다르게 나온다.  
따라서 변수 A는 “진짜 변수”이다.

하지만 수식이 “B-B+3”인 경우  
B의 값과 관계없이 계산 결과는 3이 되므로 변수 B는 “진짜 변수”가 아니다.

수식 “A+B+C”의 경우  
변수 A, B, C 각각에 대입되는 값이 변경되면 계산 결과가 변경되므로  
모두 “진짜 변수”들이 된다.  
따라서, 수식 “A+B+C”에서 “진짜 변수”의 개수는 3개가 된다.

다음은 주어진 수식 속의 “진짜 변수”의 개수를 보여준다.  
3+5 : 0개  
A+B-A : 1개  
(A+B+C)\*2-B-B : 2개  
(A+B)\*(A+C) : 3개  
(A+B)\*(C-C) : 0개  
A\*B : 2개

작성해야 하는 프로그램에서는  
수식이 주어질 때마다 이전에 주어졌던 수식들 중 동일한 수식이 몇 개인지를 찾아야 한다.

두 개 이상의 수식이 동일한지 여부는 다음과 같이 확인할 수 있다.

수식 내에 “진짜 변수”가 없는 경우 계산 결과가 같은 수식들은 동일한 수식들로 판단한다.  
아래 수식들은 “진짜 변수”가 없고 계산 결과가 13이므로 모두 같은 수식들이다.  
(3+((5)+5))  
3\*4+1-A+(A)  
(5+5-2+5)\*(B-A+1+A-B)  
(3-5-5)\*(2-4)-1

수식 내에 “진짜 변수”가 한 개 이상 있는 경우  
“진짜 변수”가 동일하게 포함되어 있고,  
“진짜 변수”들에 어떠한 정수를 대입하더라도 수식들의 계산 결과가 같으면 동일한 수식들로 판단한다.  
(계산 결과는 정수형 타입을 벗어나지 않는다고 가정한다.)

아래 수식들은 “진짜 변수”로 A를 가지고 있으며  
A 에 임의의 정수 값을 대입했을 때 모두 동일한 계산 결과가 나오기 때문에 같은 수식들이다.  
A+1+5  
A\*(C-C+2)-A+3+3  
(A+3)+A-(A-3)  
(B-3)-((B-3)-A)-(1-(3+4))

아래 수식들도 동일한 방식으로 확인해 보면 모두 같은 수식들 임을 알 수 있다.  
(A+5)\*(A+5)  
((A+4)\*(A+3+3))+1  
(A+3)\*(A+B-B+3+4)-(C-5)+(C+5)-1-5

수식의 ID와 변수들의 값이 주어지면 해당 수식에 변수 값을 대입해서 결과값을 계산 할 수 있어야 한다.  
찾을 수식의 ID가 3, A의 값이 2, B의 값이 3, C의 값이 5로 주어지는 경우  
ID가 3인 수식이 (A+2)\*(B+3)이라고 한다면 A와 B의 값을 대입하여 계산하면 24라는 결과를 얻을 수 있다.

주어지는 수식들이 위와 같은 특성을 가지고 있을 때, 동일한 수식의 개수와 계산 결과를 반환하는 프로그램을 작성하여라.

※ 수식은 연산자 우선 순위를 비교하면서 2개의 Stack에 값과 연산자를 넣어주면 계산이 가능하다.  
   다음은 3+5\*(1-5+1)+1을 Stack을 이용하여 계산한 예이다. 계산 결과는 -11 임을 알 수 있다.  
    
   쇼지이(가) 표시된 사진

자동 생성된 설명

쇼지, 낱말맞추기게임, 클립아트이(가) 표시된 사진

자동 생성된 설명

※ (A+B+C+1)\*(A+B+C+1)을 전개하면(괄호를 없애면) A2+B2+C2+2AB+2BC+2CA+2A+2B+2C+1이 된다.

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()** |
| 각 테스트 케이스의 처음에 호출된다. |
| **int addExpression(int mID, char mExpression[])** |
| 수식의 ID가 mID인 수식을 추가한 후, 주어진 수식과 동일한 수식들의 개수를 반환한다. 동일한 수식이 없는 경우 수식 내의 “진짜 변수”의 개수에 -1을 곱한 값을 반환한다. 다시말해, “진짜 변수”의 개수가 0개이면 0, 1개이면 -1, 2개이면 -2, 3개이면 -3을 반환한다. mID 값은 이 함수가 최초 호출시 1로 주어지며, 이 함수가 호출될 때마다 1씩 증가해서 주어진다. 수식은 mExpression 배열로 주어지며, 수식의 끝부분에는 ‘\0’이 들어가 있다.  ***Parameters***     mID: 수식의 ID (1 ≤ mID ≤ 5,000)     mExpression: 추가할 수식 (1 ≤ mExpression로 주어지는 문자열의 길이 ≤ 200)  ***Return***    주어진 수식과 동일한 수식들의 개수, 동일한 수식이 없는 경우 “**진짜 변수**”의 개수에 -1을 곱한 값 |
| **int calcExpression(int mID, int mA, int mB, int mC)** |
| 아이디가 mID인 수식에 변수 값을 대입한 후 계산 결과를 반환한다. 변수 ‘A’의 값은 mA, 변수 ‘B’의 값은 mB, 변수 ‘C’의 값은 mC로 주어진다. 해당 수식에 존재하지 않는 변수는 무시한다. ID가 mID인 수식은 이미 주어졌음을 보장한다.    ***Parameters***     mID: 계산에 사용할 수식의 ID (1 ≤ mID ≤ 5,000)    mA: 변수 A에 대입할 값 (1 ≤ mA ≤ 5)     mB: 변수 B에 대입할 값 (1 ≤ mB ≤ 5)     mC: 변수 C에 대입할 값 (1 ≤ mC ≤ 5)    ***Return***    변수 값을 대입한 수식의 결과 |

**[예제]**

아래에는 테스트케이스 1번을 처음부터 순서대로 보여준다.

|  |  |  |  |
| --- | --- | --- | --- |
| **순서** | **Function** | **Return** | **동일 수식의 ID** |
| 1 | init() |  |  |
| 2 | addExpression(1, "(3+((5)+5))") | **0** |  |
| 3 | addExpression(2, "A+1+5") | **-1** |  |
| 4 | addExpression(3, "3\*4+1-A+(A)") | **1** | {1} |
| 5 | addExpression(4, "(5+5-2+5)\*(B-A+1+A-B)") | **2** | {1, 3} |
| 6 | addExpression(5, "A\*(C-C+2)-A+3+3") | **1** | {2} |
| 7 | addExpression(6, "(3-5-5)\*(2-4)-1") | **3** | {1, 3, 4} |
| 8 | calcExpression(1, 1, 2, 3) | **13** |  |
| 9 | addExpression(7, "(A+3)+A-(A-3)") | **2** | {2, 5} |
| 10 | addExpression(8, "(B-3)-((B-3)-A)-(1-(3+4))") | **3** | {2, 5, 7} |
| 11 | addExpression(9, "(A+B+C+1)\*(A+B+C+1)") | **-3** |  |
| 12 | calcExpression(9, 4, 3, 2) | **100** |  |
| 13 | addExpression(10, "(A+5)\*(A+5)") | **-1** |  |
| 14 | addExpression(11, "((A+4)\*(A+3+3))+1") | **1** | {10} |
| 15 | addExpression(12, "(A+3)\*(A+B-B+3+4)-(C-5)+(C+5)-1-5") | **2** | {10, 11} |
| 16 | addExpression(13, "(A+B+C)\*(A+B+C)+A+B+C+A+B+C+1") | **1** | {9} |
| 17 | calcExpression(13, 4, 3, 2) | **100** |  |
| 18 | calcExpression(11, 2, 3, 4) | **49** |  |

ID가 1, 3, 4, 6인 수식들은 “진짜 변수”를 0개 가진 동일한 수식들이다.  
(3+((5)+5)), 3\*4+1-A+(A), (5+5-2+5)\*(B-A+1+A-B), (3-5-5)\*(2-4)-1

ID가 2, 5, 7, 8인 수식들은 “진짜 변수”를 1개 가진 동일한 수식들이다.  
A+1+5, A\*(C-C+2)-A+3+3, (A+3)+A-(A-3), (B-3)-((B-3)-A)-(1-(3+4))

ID가 9, 13인 수식들은 “진짜 변수”를 3개 가진 동일한 수식들이다.  
(A+B+C+1)\*(A+B+C+1), (A+B+C)\*(A+B+C)+A+B+C+A+B+C+1

ID가 10, 11, 12인 수식들은 “진짜 변수”를 1개 가진 동일한 수식들이다.  
(A+5)\*(A+5), ((A+4)\*(A+3+3))+1, (A+3)\*(A+B-B+3+4)-(C-5)+(C+5)-1-5

순서 12의 경우 ID가 9인 수식은 "(A+B+C+1)\*(A+B+C+1)”이며  
A에 4, B에 3, C에 2를 대입하면  
(4+3+2+1)\*(4+3+2+1)의 결과는 100이 된다.

**[제약사항]**

1. 각 테스트 케이스 시작 시 init() 함수가 호출된다.  
2. 각 테스트 케이스에서 mID는 addExpression() 함수 호출시 마다 1씩 증가하면서 주어진다.  
3. 각 테스트 케이스에서 addExpression() 함수와 calcExpression() 함수의 호출 횟수의 합은 5,000 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.