

Notice: 1:由于本OJ建立在Linux平台下，而许多题的数据在Windows下制作，请注意输入、输出语句及数据类型及范围，避免无谓的RE出现。 2:本站即将推出针对初学者的试题系统(与目前OJ是分开的，互不影响)，内容覆盖从语法入门到NOI的所有知识点，敬请关注。

4020: 未来程序·改

Time Limit: 1 Sec Memory Limit: 64 MB

Submit: 33 Solved: 17

[\[Submit\]](#)[\[Status\]](#)[\[Discuss\]](#)

Description

在 2111 年，第 128 届全国青少年信息学奥林匹克冬令营前夕，Z君找到了 2015 年，第 32 届冬令营的题目来练习。

他打开了第三题“未来程序”这道题目：

“本题是一道提交答案题，一共 10 个测试点。

对于每个测试点，你会得到一段程序的源代码和这段程序的输入。你要运行这个程序，并保存这个程序的输出。

遗憾的是这些程序都效率极其低下，无法在比赛的 5 个小时内得到输出。”

Z君想了一下，决定用 2111 年的计算机来试着运行这个题目，但是问题来了，Z君已经找不到96年前的那次比赛的测试数据了……

没有给出输入数据的提交答案题就不成其“提交答案题”之名，为了解决这个问题，Z君决定将这个题目改造成传统题。

Z君知道96年前的计算机的性能比现在差多了，所以这道题的测试数据中，输入数据的规模被设计成很小，从而，做这道题的选手只需要暴力模拟源代码的工作流程就可以通过它。

现在这道题摆到了你的面前。

本题是一道传统题，一共有 10 个测试点。

对于每个测试点，你的程序会得到一段程序的源代码和这段程序的输入。你的程序需要运行这段程序，并输出这段程序的输出。

关于给出的源代码的约定

Z君是一名C++选手。为了简化这个问题，Z君在给出的源代码中去掉了C++语言

的大量特性。从而这个源代码具有以下特点：

第一行必定为 `#include<iostream>`

>> 这个库中只会调用到对象 `cin`, `cout`、`endl`，`cin`的>>(int)方法和`cout`的<<(int)方法。这两个方法的返回值为`cin`和`cout`。

第二行必定为 `#include<cstdio>`

>> 这个库中只会调用到`putchar`方法。`putchar`方法的返回值为其唯一参数。

第三行必定为 `using namespace std;`

>> 对象`cin`的调用不再需要透过`std::cin`进行，`cout`和`endl`同理。

`int main()` 没有任何参数

所有的变量都是`int`或`int`数组（含高维数组）类型

>> 对象`cin`, `cout`,`endl`是例外，注意`putchar`的参数也是`int`类型的。我们保证在运行时这个参数的值在0~127中。

>> 在运行时，不会出现数组越界问题。

>> 没有维度的范围为1。也即，不会出现`int a[1][1][1][1][1]...`这样的情况。

>> 维度的范围直接由十进制常量给出。也即，不会出现`int a[(100+100)*2]`这样的情况。

所有的函数都是`int`类型，函数的参数只可能是`int`类型

>> 注意函数的返回值可以被丢弃。

>> 当没有显式地返回值时，返回0。

`bool`型被认为是一种特殊的`int`型

>> `==`在两个参数相同时返回1，否则返回0。

>> `!=`在两个参数相同时返回0，否则返回1。

>> `<`在第一个参数小于第二个参数时返回1，否则返回0。

>> `<=`在第一个参数小于等于第二个参数时返回1，否则返回0。

>> `>`在第一个参数大于第二个参数时返回1，否则返回0。

>> `>=`在第一个参数大于等于第二个参数时返回1，否则返回0。

>> `&&`在两个参数都不为0时返回1，否则返回0。

>> `||`在两个参数都为0时返回0，否则返回1。

>> `^`在两个参数中只有一个为0时返回1，其他时候返回0。

>> `!`在参数为0时返回1，否则返回0。

>> 由于`bool`型被`int`型取代了，因此所有的表达式都应该被完全计算：例如在表达式`(a && (b = c))`中，即使`a`已经被确定是0，仍然需要计算`(b = c)`的值，尽管无论`(b = c)`的值如何，整个表达式的值都是0。

可能用到的运算符及其优先级如下：

>> （最高）`()`, `[]`；

>> `!`, `[单目]+`, `[单目]-`；

>> `*`, `/`, `%`；

```
>> +, - ;  
>> <=, >=, <, > ;  
>> ==, != ;  
>> ^ ;  
>> && ;  
>> || ;  
>> = ; (只有这级运算符是右结合的)  
>> (最低) cout的<<与cin的>>。
```

所有int常量以十进制形式给出

Z君没有对源代码进行混淆，所以源代码是可读的，你不必担心出现大量嵌套的花括号或此类的“垃圾代码”

运行时使用的变量占用的空间的峰值不超过8MB。也即， 2^{21} 个int

调用函数的深度不会超过 10^3 层

可能出现连续赋值，例如 $a = (b = (c = 3) + 2) \% c$

>> 之前对c的赋值将会反映到之后对c的引用上

>> 赋值的返回值为赋值以后的值。

可能出现的程序流程控制语句：

```
>> if (statement) statement [else statement]
```

```
>> while (statement) statement
```

```
>> for ([statement]; [statement]; [statement]) statement
```

>> 那些作为条件的statement的返回值应当被视为bool型的。具体的来说，若返回值为0，则为false，若返回值非0，则为true。在for循环中，当第二个[statement]取空时，视为true。

空白字符只有新行符（包括\n和\r）和空格

声明变量时默认初始值为0，声明变量的同时不会进行赋值。

没有注释

所有的右花括号后没有分号

没有用来连接语句的逗号

没有函数和变量重名

Input

输入文件分为两个部分。

第一行，有一个整数N。它描述了源代码对应的输入文件program.in中包含的整数数目。

以下的N个整数构成了源代码对应的输入文件program.in。

这之后的部分构成了源代码program.cpp。

Output

输出文件是将program.cpp编译后输入program.in后所得到的输出。

Sample Input

```
2
1 2
#include
#include
using namespace std;
int main()
{int a, b; cin >> a >> b; cout
```

Sample Output

```
3
```

HINT

输入的所有program.cpp都是手打的，同时本题的std是出题人手打过的最长的程序。

测试点#1的program.cpp在附加文件中给出。

测试点#2到#4的program.cpp符合以下格式：

```
#include<iostream>
```

```
#include<cstdio>
```

```
using namespace std;
```

```
int main()

{

cout << <1> << endl;

}
```

在#2中：<1>处是一个仅包含加、减、乘、除、模运算和自然数常数的没有括号的表达式。

在#3和#4中：<1>处是一个不保证以上性质的表达式。

测试点#5中：没有除main以外的函数，并且整个程序中只有顺序结构。

测试点#6和#7中：没有除main以外的函数。

测试点#8中：所有的变量都是全局变量。

测试点#9和#10不保证任何特别的性质。

所有program.cpp都可以用MinGW GCC 4.7.2编译运行。这就是说，所有的program.cpp中都没有语法错误。然而由于编译命令的不同，直接编译得到的program.exe在运行时会因未为声明的变量和未设置返回值的函数设置0的缺省值而与标程产生不同的输出。

为了更准确地说明程序可能出现的要素，也作为提示，下面给出了一个上下文无关文法，其初始符号为PROGRAM。保证每个program.cpp都可被下面的文法生成，但是并非每个可被生成的程序都是合法的程序。

PROGRAM ::=

```
>> # include < iostream > # include < cstdio > using namespace std ;
```

FUNC_AND_VAR

FUNC_AND_VAR ::=

>> ϵ

>> int NAME (OPTPARAMS) { STATEMENTS } FUNC_AND_VAR

>> int NAME NAMES ; FUNC_AND_VAR

OPTPARAMS ::=

>> ϵ

>> int NAME PARAMS ::=

PARAMS

>> ϵ

>> , int NAME PARAMS ::=

STATEMENTS

>> ϵ

>> STATEMENT STATEMENTS ::=

STATEMENT ::=

>> EXPRESSION ;

>> { STATEMENTS }

>> int DEFINEVAR DEFINEVARS ;

>> if (EXPRESSION) STATEMENT

>> if (EXPRESSION) STATEMENT else STATEMENT

>> for (STATEMENT_IN_FOR ; OPTEXPRESSION ; STATEMENT_IN_FOR)
STATEMENT

>> while (EXPRESSION) STATEMENT

>> return EXPRESSION ;

STATEMENT_IN_FOR ::=

>> EXPRESSION

>> int DEFINEVAR DEFINEVARS

OPTEXPRESSION ::=

>> ϵ

>> EXPRESSION

EXPRESSION ::=

>> UNIT9

>> EXPRESSION << UNIT9

>> EXPRESSION >> UNIT9

UNIT0 ::=

>> INT_CONSTANT

>> UNIT0 [EXPRESSION]

>> (EXPRESSION)

```
>> NAME ( OPTARGUS )
```

```
>> >> 此处的NAME是一个函数名
```

```
>> NAME
```

```
>> >> 此处的NAME是一个变量名
```

```
>> cin
```

```
>> cout
```

```
>> endl
```

```
UNIT1 ::=
```

```
>> UNIT0
```

```
>> + UNIT1
```

```
>> - UNIT1
```

```
>> ! UNIT1
```

```
UNIT2 ::=
```

```
>> UNIT1
```

```
>> UNIT2 * UNIT1
```

```
>> UNIT2 / UNIT1
```

```
>> UNIT2 % UNIT1
```

```
UNIT3 ::=
```


>> UNIT2

>> UNIT3 + UNIT2

>> UNIT3 - UNIT2

UNIT4 ::=

>> UNIT3

>> UNIT4 < UNIT3

>> UNIT4 <= UNIT3

>> UNIT4 > UNIT3

>> UNIT4 >= UNIT3

UNIT5 ::=

>> UNIT4

>> UNIT5 == UNIT4

>> UNIT5 != UNIT4

UNIT6 ::=

>> UNIT5

>> UNIT6 ^ UNIT5

UNIT7 ::=

>> UNIT6

>> UNIT7 && UNIT6

UNIT8 ::=

>> UNIT7

>> UNIT8 || UNIT7

UNIT9 ::=

>> UNIT8

>> UNIT8 = UNIT9

OPTARGUS ::=

>> ϵ

>> EXPRESSION ARGUS

ARGUS ::=

>> ϵ

>> , EXPRESSION ARGUS

DEFINEVARS ::=

>> ϵ

>> , DEFINEVAR DEFINEVARS

DEFINEVAR ::=

>> NAME

>> DEFINEVAR [INT_CONSTANT]

NAME

>> 一个[A-Za-z_]中的元素加零或一或多个[0-9A-Za-z_]中的元素。

INT_CONSTANT

>> 一个[1-9]中的元素加零或一或多个[0-9]中的元素。

Source

2015年集训队互测

[\[Submit\]](#)[\[Status\]](#)[\[Discuss\]](#)

[HOME](#) [Back](#)

[한국어](#) [中文](#) [فارسی](#) [English](#) [ไทย](#)

版权所有 ©2008-2012 大视野在线测评 | 湘ICP备13009380号 | 站长统计

Based on opensource project hustoj.