**128MB,1S,.xxx**

下标,内存和字符串的狂拽

**问题描述**

**本题并不需要用到指针**

在V国和J国的边境处，发现了一个文艺复兴时代的工坊，在此中，发现了一个过时无效的历史程序，它无法在任何现代机器上执行，但是为了发掘文化优越性，V国带桶勇下令成立研究专家组。尽管V国和J国的关系很差，但这次的单子是从J国转包过来的（由于以以往的经验，直接找的V国的承包商并不靠谱），当然因为缺乏文化自信，研究专家组先找了个V国皮包公司来保证與情稳定。

你只被分包到了其中很小的一部分，是一段代码，有$ n$个字符数组，标号$ 0$到$ n-1$, 第i个数组的大小为$ m\_i$,

char $ s\_0[m\_0], s\_1[m\_1], s\_2[m\_2], ..., s\_{n-1}[m\_{n-1}]$;

其中$ s\_i[m\_i] $的下标访问范围为 $s\_i[0] ~ s\_i[m\_i - 1]$

这些数组依次存储在内存中的一段连续无间隔的地址中。（更多信息见提示）

以下为我们要实现的功能，给定n和$m\_i$, 以及初始时各个char数组的值。

依次执行以下函数。若出现了对非法内存的访问（即访问了除了这n个数组之外的地址），应当**拒绝执行**并输出"overflow"并换行, 若没有访问非法内存，也没有满足strcpy的交集判断，但进行了越界访问，应当正常执行，但在输出之前输出一个英文字母'w'(不含引号)，'w'之后也不需要空格/换行，直接和该函数的输出连接。

越界访问，指访问了指定的数组之外的数组，具体见解释。

以下三个函数。**这里的pointer可以用数组下标模拟，详见解释**

memset(pointer start, value, count);

从start（含）开始，依次count个byte，均赋值为value。若成功输出“true”（不含引号），并换行。（若成功执行，则修改会被实际执行，即之后的操作以执行后的各数组为基础进行）

strcpy(pointer source, pointer destination);

将从source开始的字符串，粘贴到destination处，需要粘贴'\0'。若成功输出"true"（不含引号），并换行。若粘贴过程中，source字符串的位置和destination对应的位置中存在交集，并且没有进行过非法访问，则应当拒绝执行并输出“overwrite”。若进行过非法访问，则无需输出"overwrite"。

strlen(pointer start);

统计从source开始的字符串的长度，需要访问'\0', 但并不计入数值。若成功输出长度，并换行。

**输入描述**

第一行一个整数$ n$ ,表示数组个数，

接下来一行，$ n$ 个整数$ m\_i$，依次表示第$ 0$ 到第$ n-1$ 个数组的大小

接下来$ n$ 行，每行$m\_i$个整数$v\_{i,j}$，依次表示第i个数组的从$ 0$ 到$m\_i$的位置上的值

接下来一行一个整数$ k$ ，表示要执行多少次函数

每行若干个整数格式如下（用一个空格隔开，没有逗号）**，&的含义详见解释**

$ 0,s\_x,s\_y,v,c表示memset(\&s\_{s\_x}[s\_y], v, c); $

$ 1,s\_{x0},s\_{y0},s\_{x1},s\_{y1}$

$ 表示strcpy(\&s\_{s\_{x0}}[s\_{y0}], \&s\_{s\_{x1}}[s\_{y1}]);$

$ 2,s\_{x0},s\_{y0}$

$ 表示strlen(\&s\_{s\_{x0}}[s\_{y0}]);$

$ 1 <= n, m, n \* m <= 1000; 0 <= s\_x <=n - 1 ; 0 <= |s\_{y}|,c <=10^9; v \in \{0, 1\}; 1 <= k <= 5\*10^4$

输入均符合格式要求

**输出描述**

对于每次函数执行，输出一行，内容见题目描述。

**样例输入**

3

3 2 4

1 0 1

1 1

1 0 0 1

19

0 2 3 1 1

0 2 4 1 0

0 2 4 1 1

0 1 2 1 1

2 0 1

2 0 0

2 0 1

2 0 3

2 0 5

2 1 -1

2 2 2

2 2 3

2 0 5

1 0 1 2 100

1 0 1 2 0

2 1 1

1 0 2 1 0

2 0 2

1 1 1 2 2

**样例输出**

true

true

overflow

wtrue

w4

0

1

0

w3

w1

w4

0

overflow

w1

overflow

true

w1

overwrite

w3

wtrue

**数据范围及提示**

这些数组将被存储在内存中，内存本身可以看作由若干个{8bit组成的byte} 构成的数组A，~~而指针就相当于这个大数组的下标~~。$ s\_0$到$ s\_{n-1} $依次连续存储在这个大数组中，（当然通常不会从0开始）我们设$ s\_0[0]$存储在$ A[P]$, 则$s\_0[1]$存储在 $A[P+1]$, $ s\_1[5]$ 存储在 $A[P+m\_0+5]$，那么$ s\_x[y]$存储在哪里呢？

我们其实可以把$ s\_0$看作一个下标P，$s\_0[x]$可以等价的写为$ A[s\_0+x]$,

那么$s\_0[m\_0]$就是$A[s\_0+m\_0]$, 也就是$s\_1[0]$, 也是$A[s\_1]$。

注意类似$s\_0[m\_0]$和$s\_1[-1]$这样的访问是可以执行的，但在本题中应该提出warning（而并不是认为不可执行）。

如果n很大，依次求和可能会导致时间复杂度过高（当然本题不会有这个问题），可以通过计算一个前缀和数组，来简化指针计算。

**我们在此定义&是一个单目运算符（C中的&有类似但不相同的含义）**，表示取出对象在A中的坐标，比如$&s\_2[3] == s\_2 + 3$

$&s\_5[7] = s\_0+n\_0+n\_1+n\_2 + n\_3 + n\_4+7$

我们发现，可以处理一个数组sum,满足

$sum[i] = s\_0+\displaystyle\sum\_{j=0}^{i-1} $

同时有sum[i] = s\_i

我们可以这样计算

sum[0] = s\_0;

for (int i = 1; i <= n; ++i) {

sum[i] = sum[i - 1] + $n\_{i-1}$;

}

sum[n]表示的是访问的最大坐标-1

计算$&s\_5[7]$时，可以直接$sum[5]+7$

由于我们本题并没有对$s\_0=P$有任何约定，所以你可以直接开一个数组,

$char A[maxn];//\displaystyle\sum\_{i=0}^{n-1}n\_i$

令s\_0=A;

字符类型char，其实等同于有符号的8bit整数，其编码规则与int相同采用补码。ASCII码其实就是在定义域内将数值的含义和数值进行的一一映射。char数组存储字符串会以'\0'即数值0为结尾标志，这个标志在输入输出时并通常不会显式出现，但在处理字符串时，这个标志位就很重要了，因为它将告诉你何处该字符串结束。（虽然不必要，但在自己在一后的编程中，如果使用char数组，也可以考虑采用'\0'作为字符串标志位）

本题的越界访问不需要考虑被拒绝执行的情况，以下几种情况符合本题的非法越界

（1）$ n\_0 == 1$但访问了$s\_0[1]$（无论是运行当中，还是直接作为参数传入并进行了至少一次对该处访问）

（2）$ strcpy(s\_0, s\_1 + 5)$，若在粘贴过程中，只访问了$ s\_0，s\_1$ ，但source串的位置包括$ s\_1[0]$,哪怕只是最后的'\0',也属于越界。

若$strcpy(s\_1+5, s\_0)$的destination串的位置包含了$s\_1[0]$也属于越界

（3）$ s\_1-2$作为参数传入并进行了至少一次对该处的访问时，就已经越界了

特别地，如果不需要访问，哪怕直接传入一个非法地址或者越界地址，也不需要认为其发生了非法访问/越界。比如$memset(&x\_0[-100], 0, 0);$

区分“true”和“ture”