single 专题

一.　　题目概况

英文题目	prime	fork	maze
输入文件名	prime.in	fork.in	maze.in
输出文件名	prime.out	fork.out	maze.out
每个测试点时限	1s	1s	1s
测试点数目	10	10	5
每个测试点分值	10	10	20
结果比较方式	全文比较（过滤行末空格及文末回车）		
题目类型	传统	传统	传统
运行内存上限	256MB	256MB	256MB

1. single 找素数

【问题描述】

single 想找出区间[x,y]中的素数个数，完。

【输入格式】

输入共一行，两个正整数 x 和 y 表示区间[x,y]

【输出格式】

输出共一行，一个数，表示区间[x,y]中的素数个数

【输入输出样例】

输入样例#1：	输出样例#1：
2 11	5

【数据范围】

对于 20%的数据：1 <= x <= y <= 10^6

对于 100%的数据： 1 <= x <= y <= 2147483647, y-x<=1000000

2. single 的自我复制

【问题描述】

众所周知，single 会自我复制。

他的能力来源于 fork，是他体内的一个内置函数。如果 single 调用这个函数，那么他会复制出一个**完全一模一样**的自己（**包括之前的运行状态**），并且自己的函数返回 1，复制出来的自己的函数返回 0，然后两只 single 都继续执行程序。

初始时只有一只 single。接着，LY 让 single 计算这样的表达式：

fork() <op> fork() <op> … <op> fork()

其中 <op> 是二元运算符，为 && 或者 || 中的一种。例如：

fork() && forrk() || fork() && fork() && fork() || fork()

两个运算都是左结合，且 && 的优先级比 || 高，所以上面的那个表达式相当于：

((fork() && fork()) || ((fork() && fork()) && fork())) || fork()

对于表达式 a && b，single 会先计算 a 的值，如果为 0 那么**不计算** b 的值（因为很重要所以说两遍，请注意这里**不计算** b 的值），该表达式值为 0；否则计算 b 的值并将其值作为该表达式的值。

对于表达式 a || b，single 会先计算 a 的值，如果为 1 那么**不计算** b 的值（因为很重要所以说两遍，请注意这里**不计算** b 的值），该表达式值为 1；否则计算 b 的值并将其值作为该表达式的值。

很多年后，single 的后代（**非亲生**）得到了 single 的表达式，想知道 single 当年究竟复制了多少个自己。

你可以参照样例来更好的理解题意。

【输入格式】

第一行一个正整数 n，表示表达式中的 fork() 的数量。

接下来一行 $n - 1$ 个用空格隔开的字符串，每个字符串为 “&&” 或者 “||”，依次表示表达式中对应位置的运算符。

【输出格式】

一行，一个整数表示复制出 single 的数量，你只用输出答案对 998244353 取模后的结果。

【输入输出样例】

输入样例#1	输出样例#1
2 &&	3
输入样例#2	输出样例#2
6 && || && && ||	15

【数据范围】

对于 10%的数据：n == 1

对于 30%的数据，n <= 5

对于 60%的数据，n <= 100

对于 100%的数据，n <= 100000

【说明】

样例 1 的说明：

共生产 3 只 single，过程如下：

1. 第 1 只 single 开始计算 fork() && fork()。
2. 第 1 只 single 开始计算 fork()，产生了第 2 只 single。
3. 第 1 只和第 2 只的 fork() 计算完成，第 1 个返回 1，第 2 个返回 0。
4. 第 1 只 single 由于 fork() 返回值为 1，开始计算 fork() && fork() 右边的 fork()，产生了第 3 只 single。
5. 第 2 只 single 由于 fork() 返回值为 0，于是 fork() && fork() 值为 0（跳过右边的 fork 的计算），程序结束。
6. 第 1 只和第 3 只的 fork() 计算完成，第 1 个返回 1，第 3 个返回 0。
7. 第 1 只 single 由于 fork() 返回值为 1，于是 fork() && fork() 值为 1，程序结束。
8. 第 3 只 single 由于 fork() 返回值为 0，于是 fork() && fork() 值为 0，程序结束。

3.single 走迷宫

【问题描述】

single 被困在了迷宫中，他现在要像老鼠走迷宫那样走出去。

迷宫是一个 N * M 的矩阵，每个位置可以用（行号 x，列号 y）来表示，两个位置间有可能是一扇锁着的门，或是一堵不可逾越的墙。迷宫中的一些位置放着钥匙，所有门被分成 P 类，打开同一类的门的钥匙相同，不同类的门的钥匙不同。

single 被困在西北角（1，1），他需要移动到出口，也就是东南角（N，M），single 只能上下左右移动，且从一个位置移动到另一个位置所花费的时间为 1s，拿钥匙和开门的时间忽略不计。

求出 single 到达目标点的最短时间。

【输入描述】

第一行 3 个整数，分别是 N，M，P。

第 2 行是 1 个整数 K，表示迷宫中门和墙的总数。

第 I+2 行（1<=I<=K），有 5 个整数，依次为 Xi1,Yi1,Xi2,Yi2,Gi：

当 Gi>=1 时，表示(Xi1,Yi1)单元与(Xi2,Yi2)单元之间有一扇第 Gi 类的门，

当 Gi=0 时，表示(Xi1,Yi1)单元与(Xi2,Yi2)单元之间有一堵不可逾越的墙（其中，|Xi1-Xi2|+|Yi1-Yi2|=1，0<=Gi<=P）。

第 K+3 行是一个整数 S，表示迷宫中存放的钥匙总数。

第 K+3+J 行(1<=J<=S),有 3 个整数，为 Xi1,Yi1,Qi:表示第 J 把钥匙在(Xi1,Yi1)单元里，且第 J 把钥匙用来开第 Qi 类门。（其中 1<=Qi<=P）

输入数据中同一行各相邻整数之间用一个空格分隔。

【输出描述】

输出最短时间，若无解则输出 -1

【样例输入输出】

样例输入#1	样例输出#1
4 4 9 9 1 2 1 3 2 1 2 2 2 0 2 1 2 2 0 2 1 3 1 0 2 3 3 3 0 2 4 3 4 1 3 2 3 3 0 3 3 4 3 0 4 3 4 4 0 2 2 1 2 4 2 1	14

【数据范围】

对于 100%的数据，1 <= N,M,P <= 10