Problem I
Oracle

Time limit: 3 seconds
Memory limit: 256 megabytes

Problem Description

Oracle is a concept in computer science that serves as a black box which can solve some sort of problem in a single operation. It is a tool used to study decision problems. The origin of oracle goes back to ancient Greece. The most famous of them is probably Pythia, the oracle of Delphi, priestess to Apollo. Pythia is not the name of one person, but a title given to the oracle of Delphi. They are active and continued to be consulted by Greeks from 7 century BC until the 4th century AD. And Phemonoe, the first oracle of Delphi, possibly the daughter of Apollo himself, is the main character in this problem. Why, you might ask, does this problem not feature a computer scientist whose name started with I? Well we simply cannot find a computer scientist whose name started with I that has anything to do with this problem, so get over it.

Phemonoe is in a series of n games. The games happen sequentially. You cannot go back to play in a previous game that is already done, as time travel is not in the realm of Apollo's powers. She can choose not to participate in a game though. In these games, she can place bets on each of them and receive money that is some multiple of the bet she placed. Usually you shouldn't know the result of each game. But since she is an oracle of Delphi, Apollo already told her what would happen exactly. However, Apollo did warn her to not be too greedy. Being too suspicious and win too much is certainly not a good idea. Zeus might not be so happy if he finds out what Apollo did, you see. Unfortunately, after a few glasses of good quality Greek wine, Phemonoe's judgment is not be as good as she needs it to be. As her confidence grows, she also starts to raise her bets. As her adviser, your job is to maximize her profit. Note that she will not listen to you in terms of the total number of games she wants to play and the amount she is going to bet, but she will listen to your advice as to which games to participate.

Input Format

On the first line, there is a single integer T ($T \leq 20$) indicating the number of test cases. For every test case, the first line will contain a single integer representing the total number of games n ($n \leq 50000$). On the second line, there are n integers $p_1, p_2, \ldots, p_n$, representing the multiplier for each game. Phemonoe's winning of each game will be the product of her bet and the multiplier. Negative multipliers mean losing. For every $i \in \{1, \ldots, n\}$, you may assume $|p_i| \leq 50000$. The third line contains two non-negative integers a and b ($a, b \leq 100$), indicating that Phemonoe will bet $j^2 + aj + b$ where $a^2 \geq 4b$ on the j-th game she participates.

Output Format

For every test cases. Output n integers on a line. Representing the maximum profit if Phemonoe decided to participate in $1, 2, \ldots, n$ games respectively.

Sample Input

```
2
3
1 2 3
0 0
5
1 -1 1 -1 1
3 2
```

Sample Output

```
3 14 36
6 18 38 44 26
```

2016 ACM-ICPC Asia Taiwan Online Programming Contest
競賽資訊

競賽規則

違反下列規則，將導致參賽者失去參賽資格。

一、 不得使用任何機器可讀的資料，如預先寫好存放於電腦中的程式碼。但可以使用紙本資料，如教科書、字典、筆記以及列印好的紙本程式碼。

二、 在比賽過程中，參賽者只能與隊友討論。競賽期間與教練或其他隊伍聯繫均屬違規行為。

三、 參賽者只能夠透過網路下載題目敘述、上傳解答程式碼、提問澄清疑點與查看計分板。使用網路存取其他資訊均屬違規。

四、 每個隊伍僅可使用一台電腦撰寫程式與上傳程式碼。於競賽期間除使用印表機列印題目與程式碼以及透過額外的螢幕閱讀題目之外，不得使用任何其他電子裝置。

五、 不得做出任何意圖妨礙比賽進行及影響比賽公平性的惡意行為。

賽後教練需上傳一張競賽時照片並保證該隊伍遵照規則進行比賽。

計分與排名

一、 違反競賽規則以致失去參賽資格者，不予計分與排名。

二、 本次競賽僅提供 C、C++、Java、Python ，不接受其他程式語言。程式需能正常執行，且在時間限制內計算完畢輸出正確答案，才算答對。以下為裁判系統常見回應：

- CE: 語法有錯誤或其他因素以致於無法編譯或執行。
- TLE: 未能在時間限制內執行完畢。
- RE: 執行時錯誤，程式無法正常執行完畢。
- MLE: 未能在記憶體限制內執行完畢。
- WA: 在時限內執行完畢但輸出錯誤。
- AC: 答對。

三、 隊伍以解題數量多者排名較前，解題數量相同時，以總消耗時間少者排名較前。答對的題目的消耗時間計算方式為比賽開始至解出題目所消耗的分鐘數。如解出前有答錯，每答錯一次需要另加 20 分鐘。總消耗時間為所有答對題目的消耗時間加總。未答對的題目不計消耗時間。

四、 如果在前述計分規則無法分出勝負，則依序以下列規則決定排名：

(a) 總提交次出少者排名較前
(b) 最先解出的題目消耗時間較短者
(c) 次先解出的題目消耗時間較短者，如再相同則類推

Problem A
Alan Turing

Time limit: 1 second
Memory limit: 256 megabytes

Problem Description

艾倫・麥席森・圖靈 (Alan Mathison Turing) 是位偉大的英國際算機科學家、數學家。作為研究計算機科學的先鋒，提出了一個影響重大的數學抽象模型：圖靈機。

以下將簡單介紹一下圖靈機是如何運作的。圖靈機有一定數量的狀態、一條無窮長且能紀錄特定符號的帶子，依據一組特定的規則進行狀態間的轉移。起始時，圖靈機會處於一個特定的起始狀態，其讀寫頭會指在一個特定的位置。圖靈機會依據當下的狀態以及讀寫頭位置上的符號，來決定接下來將往哪一個方向移動讀寫頭或是停機、要將什麼符號寫回資料帶上、以及下一個狀態將會是哪一個。

「有沒有辦法判斷特定的圖靈機在給定特定的資料帶時，最終會停機還是無止境的持續執行？」便是所謂的「停機問題」，是圖靈機相關理論中的一個有趣問題，可已經被證明是無法判斷的了。然而，停機問題仍有些正面的結果。「特定圖靈機在給定特定的資料帶時，是否能在 10 步內停機」是完全可以判斷的。這個版本的問題，就請你寫個程式來回答吧！

Input Format

測資輸入的第一行，有一個整數 T ($T \leq 20$) 代表測試資料的筆數。

每一筆測試資料會先描述圖靈機。測試資料的第一行，會有一個整數 n ($1 \leq n \leq 10$) 代表這臺圖靈機有 n 個狀態。其中狀態 1 是起始狀態，而狀態 n 為唯一的停止狀態：一進入該狀態，圖靈機便停機。

接下來的 $n-1$ 行會依序描述圖靈機狀態 $1, \ldots, n-1$ 的轉移規則。請注意狀態 n 為停止狀態，因此不會有任何的狀態轉移規則。為簡化這個問題，在資料帶上會出現的符號僅有 $0, 1, 2$ 三種。我們將使用三個以一個空白隔開三元組 (3-tuple) 描述轉移規則。每個三元組 (x, y, z) 代表圖靈機將轉移到狀態 x 、$y = 1$ 時讀寫頭將往右移、$y = -1$ 時讀寫頭將往左移、z 則是寫回資料帶的符號。在本題的設定中，圖靈機將先將符號寫回資料帶，再進行讀寫頭的移動。這 $n-1$ 行中的第 i 行的三個三元組，分別代表了在狀態 i 讀寫頭讀到符號 0、i、2 的三種狀態轉移。舉例來說，第五行的第二個三元組如果是狀態 5 讀到符號 1 時該進行的狀態轉移。

接下來的測試資料會有一個整數 m 代表這組測試資料有 m 筆查詢。每筆查詢會有幾個數字寫在單獨一行。其中第一個數字 x ($0 \leq x \leq 10$) 代表這筆查詢有連續 x 個符號預先寫在資料帶上。接下來共有 x 個數字，就代表寫在讀寫頭起始位置起，往右方延續的符號。這 x 個數字只會有 0 跟 1，符號 2 其實是空白。因此當輸入為 `3 1 0 0` 時，其實資料帶看起來是 `... 2 2 2 1 0 0 2 2 2 ...` ，且初始時讀寫頭指在 `1` 的位置，在開頭前方與結尾後方都有無窮多個空白 (符號 2)。

Output Format

對第 N 組測試資料請先印出 `Machine #N:`。隊每一筆查詢，如果能在 10 步內停機，則印出 `yes` 否則印出 `no`。

Sample Input

```
3
3
(2, 1, 0) (2, 1, 1) (2, 1, 2)
(3, 1, 0) (1, -1, 0) (1, -1, 2)
3
3 1 0 0
2 1 1
1 1
4
(2, 1, 1) (3, 1, 0) (1, -1, 2)
(1, -1, 0) (1, -1, 1) (1, -1, 2)
(2, 1, 1) (2, 1, 0) (4, -1, 2)
2
3 0 0 0
5 0 0 0 0 0
2
(1, -1, 2) (2, -1, 0) (1, 1, 2)
2
2 0 0
1 1
```

Sample Output

```
Machine #1:
yes
yes
no
Machine #2:
yes
no
Machine #3:
no
yes
```

Problem B
László Babai
Time limit: 1 second
Memory limit: 256 megabytes

Problem Description

> László Babai 是位匈牙利計算機科學家以及數學家。他曾獲得哥德爾獎 (Gödel prize)，在計算理論、演算法、組合數學與群論等領域，他都有傑出的研究成果。去年，他提出了一個能在次指數時間內解決圖形同構問題的演算法。在那之前，最佳的結果是一個 $\exp(O(\sqrt{n \log n}))$ 時間的演算法，而 Babai 的新結果推進到了 $\exp((\log n)^{O(1)})$。

圖形同構 (Graph Isomorphism) 是計算機科學領域中的一個著名 NP 問題。或許你不知道那是什麼，讓我簡單解釋給你聽。給定兩個圖形 $A = (V_A, E_A)$ 、$B = (V_B, E_B)$，其中 A 的點集合 $V_A = \{a_1, a_2, a_3, \ldots, a_{n_A}\}$ 且 B 的點集合 $V_B = \{b_1, b_2, b_3, \ldots, b_{n_B}\}$。$A$ 和 B「同構」等價於下列兩個條件同時成立：

1. A 和 B 有相同數量的點與邊。
2. 存在一個對射 (一對一且映成) 函數 $f : V_A \to V_B$ 使得 $\{u, v\} \in E_A$ 若且唯若 $\{f(u), f(v)\} \in E_B$.

換句話說，我們能夠將 A 的點集重新編號，便可做出 B 來。

圖形同構問題仍然還不知道是否為 NP-complete 。作為下一代的電腦科學家，我們必須胸有大志，永遠不怕發大夢！就讓我們挑戰一下，判斷兩個僅有三點的無向簡單圖形，是否同構吧！告訴這個世界，我們也能有所成就！

Input Format

測試資料的第一行有一個整數 T ($T \leq 100$) 代表有 T 筆測試資料。

每一筆測試資料會以一個整數 m ($0 \leq m \leq 3$) 開始，代表圖形 A 有 m 條邊。接下來 m 行，每一行都有兩個相異整數 u, v ($u \neq v$, $u, v \in \{1, 2, 3\}$) 來描述有一條邊連接 u, v 兩個點。你可以假定任兩點間最多只有一條邊。測試資料中，剩餘的部份用相同的格式描述圖形 B。

Output Format

對每一筆測試資料，如果兩圖形同構，則輸出 `yes` ，否則輸出 `no` 。

Sample Input

```
3
3
1 2
2 3
3 1
3
1 3
2 1
3 2
2
1 2
1 3
0
1
2 3
1
1 2
```

Sample Output

```
yes
no
yes
```

Problem C
Stephen Cook

Time limit: 3 seconds
Memory limit: 256 megabytes

Problem Description

> 史提芬・亞瑟・庫克 (Stephen Arthur Cook) 是個計算機科學家，專長領域是艱澀的計算複雜度理論。他在論文 "The complexity of theorem-proving procedures" 中提出了精確的 P 對 NP 問題的經確定義，而他最重要的貢獻之一，是證明了 Boolean satisfiability (SAT) 問題為 NP-hard。

庫克為 1982 年圖靈獎得主。是的，就是 Problem A 的那位圖靈。恰好你也是個叫做庫克的宅宅，你的宅宅好朋友列文 (Levin) 跟你挑戰一起玩個由庫克定理啟發宅宅的遊戲。這遊戲是這樣進行的，首先你們準備了一個 Boolean formula (實際上是出題者準備的)，然後你跟列文輪流，每一個回合，挑選一個變數 (variable) 並決定其真假值。直到所有的變數值被設定時，遊戲就結束。如果最後 Boolean formula 運算出來的值為真，則你獲勝，反之列文獲勝。假定你先手，且列文總是用最佳策略進行遊戲。你有宅到可以告訴我，究竟誰會勝利嗎？

Input Format

輸入資料的第一行有一個整數 T ($T \leq 20$) 代表有多少組測試資料。

每一組測試資料有兩行，第一行有一個整數 n ($1 \leq n \leq 10$) 代表 Boolean formula 有多少個變數，其中第 i 個變數會用第 i 個英文字母的大寫代表。即第一個變數為`A`、第三個變數為`C`。

第二行有一個至多有 256 個字元的 Boolean formula。一個 Boolean fomula 必須為下列五種形式之一：

- `var`: `var` 是一個變數。
- `( formula1 )`: `formula1` 是一個 Boolean formula。
- `not formula1`: `formula1` 是一個 Boolean formula。
- `formula1 or formula2`: `formula1` 和 `formula2` 都是 Boolean formula。
- `formula1 and formula2`: `formula1` 和 `formula2` 都是 Boolean formula。

你可以假定變數跟運算子之間都有空白隔開。

本題只有似種運算子：`and`、`or`、`not`、括號`()`。前三種都用小寫字母。運算優先順序為：`()` > `not` > `and` > `or`。

Output Format

請輸出一行是誰贏了。請輸出 "`Cook`" 或 "`Levin`"。

Sample Input

```
3
1
A and not A
1
A or not A
3
( A or C ) and B
```

Sample Output

```
Levin
Cook
Cook
```

Problem D
Edsger Dijkstra

Time limit: 2 seconds
Memory limit: 256 megabytes

Problem Description

圖靈獎得主艾茲赫爾・韋伯・戴克斯特拉 (Edsger Wybe Dijkstra) 是荷蘭計算機科學家，他的名字的發音總是讓許多亞洲人感到困擾。在 1960 年代，他給出了以下的評論：

> For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce. More recently I discovered why the use of the go to statement has such disastrous effects, and I became convinced that the go to statement should be abolished from all "higher level" programming languages.

你不想產出低品質的程式碼，對吧？然而你發現你的程式碼裡面一個迴圈敘述都沒有。唯一的流程控制程式碼，就只有`if-goto`。為了降低`goto` 的密度，你得試著消除掉一份類 C 程式碼裡所有的`if-goto` 敘述，用`do-while` 迴圈以下面的方式取代：

- 假定`if-goto` 看起來像是 "`if (boolean_expression) goto some_label;`".
- 在`some_label` 宣告處後方插入一份 "`do {`"。
- 用 '`} while`' 取代 '`if`'。
- 移除 '`goto some_label`'。

舉例來說，就是將以下程式碼

```
int main() {
    int score;
    get_score:
    scanf("%d",&score);
    if (score < 0 || score > 100) goto get_score;
    if (score < 60) goto fail;
    fail:
    puts("you are failed!");
    return 0;
}
```

轉換為下面這樣：

```
int main() {
int score;
    get_score: do {
    scanf("%d",&score);
    } while (score < 0 || score > 100) ;
    } while (score < 60) ;
    fail: do {
    puts("you are failed!");
    return 0;
}
```

以上的程式碼無法通過編譯，並不是一個令人意外的結果。你的新任務是：給定一連串的敘述，請判斷是否所有的`if-goto` 敘述，可以用`do-while` 迴圈，以前述的方式取代，並且不影響到程式的輸出。為了簡單起見，你可以假定程式碼中的敘述只有下列兩種格式：

- 格式一：“`line_`x`: puts("`x`");`” 其中 x 是此敘述對應的行號且 “`puts("`x`");`” 會印出行號 x。
- 格式二：“`if (expr_`x`()) goto line_`y`;`” 其中 x 是該敘述對應的行號，`line_`y 是程式中合法的標籤 (Label)。“`expr_`x`()`” 在前 x 次呼叫時，將會回傳 `true`，之後會回傳 `false`。

請留意如果程式碼無法通過編譯，將被視為與原先的程式碼輸出不同。

Input Format

輸入資料的第一行有一個整數 T ($T \leq 20$) 代表有多少組測試資料。每一組測試資料由一連串的敘述所構成，每行有一個敘述。而構成測試資料的敘述有三種：格式一、格式二以及`END`。`END` 代表該組測試資料結束，只會出現在每筆測試資料的最後一行，不用把它看作程式碼的一部分。單一筆測試資料，至多有 10000 個敘述 (不含`END`)。

Output Format

如果將所有的`if-goto` 敘述換成`do-while` 迴圈不會改變輸出的話，請印出 “good”，反之印出 “bad”。如果程式碼將無法編譯，請輸出 “bad”。

Sample Input

```
3
line_1: puts("1");
if (expr_2()) goto line_1;
END
line_1: puts("1");
line_2: puts("2");
line_3: puts("3");
if (expr_4()) goto line_2;
line_5: puts("5");
if (expr_6()) goto line_2;
END
line_1: puts("1");
if (expr_2()) goto line_5;
line_3: puts("3");
line_4: puts("4");
line_5: puts("5");
END
```

Sample Output

```
good
good
bad
```

Problem E
Jack Edmonds

Time limit: 2 seconds
Memory limit: 256 megabytes

Problem Description

> 傑克・愛德蒙 (Jack Edmonds) 是為美國計算機科學家。或許他最為 ACM-ICPC 社群所知研究成果的是在 $O(|V||E|^2)$ 時間內解決網路最大流量問題的 Edmonds-Karp 演算法，但他最重大的影響可能是 Cobhan-Edmonds 假說：用多項式時間來定義演算法能被實踐。如今我們把決定性多項式時間 (P) 當作是可以有效率「解答」的問題類別，而非決定性多項式時間 (NP) 當作是可有效率「驗證答案」正確性的類別。所謂的 P 對 NP 問題，就是在問「這兩個問題類別，是否相等?」，這是目前計算機科學界尚未知曉答案的問題，也是七個千禧年大獎之一。多數的計算機科學家相信 $P \neq NP$，但仍缺乏正確性的證明。另一方面，許多人試著利用證明「特定 NP-hard 問題 (至少跟任一 NP 問題一樣難) 可在多項式時間內解答」來推論 $P = NP$，但也都失敗告終。

旅行推銷員問題，是一個知名的 NP-hard 問題。該問題是：「給定一個目的地清單以及任兩個目的地之間的最短距離，要如何從一個起點拜訪所有目的地，並回到起點」。我相信像你這樣優秀的程式設計失，你一定可以很快搞定這個小問題。為了讓這問題困難點，我們將不提供你地圖，你只會得到每個目的地所在的座標點，而且一開始這些目的地之間，還沒有興建任何道路。顯然的，當有一些目的地與其他目的地連通時，你無法拜訪所有的目的地。

假定有 n 個目的地，座標分別為 $(x_1, y_1), \cdots, (x_n, y_n)$ 且 (x_1, y_1) 是起點。市長願意協助你，替你興建一些道路，讓你可以拜訪所有目的地。然而，市長又笨又懶。因為笨，他只能興建由垂直與水平路段所構成，直接連通兩個目的地 (x_i, y_i)、(x_j, y_j) 的道路。這道路的長度為 $|x_i - x_j| + |y_i - y_j|$。因為懶，他只會替你興建 $n - 1$ 道路，他的顧問告訴他，$n - 1$ 條道路就足以讓你到達所有目的地。你可以決定興建哪些道路，請你告訴我從起點出發，拜訪所有目的地後回到原點的最短路徑長度是多少。

Input Format

測試輸入的第一行有一個整數 T $(T \leq 20)$ 代表有多少組測試資料。每一組測試資料的第一行，會有一整數 n $(n \leq 10,000)$ 代表有多少目的地。接下來的 n 行，每一行都會有兩個整數 x, y $(-1,000 \leq x, y \leq 1,000)$，代表對應的目的地之座標。

Output Format

對每一組測試資料，印出從起點出發，拜訪所有目的地後回到原點的最短路徑長度是多少。

Sample Input

```
3
3
1 1
2 2
3 3
4
2 1
-1 2
-2 -1
1 -2
6
1 2
2 3
2 2
3 4
4 3
3 1
```

Sample Output

```
8
24
16
```

Problem F
Robert Floyd

Time limit: 1.2 seconds
Memory limit: 256 megabytes

Problem Description

羅伯特・W・弗洛伊德 (Robert W Floyd) 是個美國計算機科學家，也是個圖靈獎得主。在 ACM-ICPC 社群中，他最為人知的成果，可能是 Floyd-Warshall 演算法，可用於求出任兩點間最短路徑以及傳遞閉包 (Transitive closure)。此演算法有相當多應用，但或許無法用於解決本問題。

縫合怪引起了夜色鎮的恐慌。縫合怪會沿著行進路徑分泌出腐敗的膽汁，不但又臭又噁心還會讓你在上面行走的移動速度下降 35%。簡而言之，你不想要踩在那上面。隨著縫合怪的移動，他可能將整個地圖切成好幾個被分開的區域。兩個區域被分開的意思是說，如果有人要從一個區域走到另外一個，非得踩過縫合怪的膽汁不可。

縫合怪總是沿著單一方向前進 (上下左右)，而且一步只走一單位長，走完之後再來決定他要不要改變他行進的方向。為了帶給夜色鎮平靜的生活，英雄們將盡快的殺死縫合怪。可以假定縫合怪最多流出 2048 單位長的膽汁，因為英雄們不會允許他在夜色鎮走更多步了。

但不幸的，縫合怪的臭膽汁並不會隨著他的死亡而消失。作為夜色鎮的鎮長，你想要算出夜色鎮被切成多少個分開的區域。我們該怎樣用 Floyd-Warshall 演算法解決這個問題？又該如何得出這個問題用 Floyd-Warshall 演算法沒有太大幫助的結論呢？因為縫合怪行進的方式，我們可以想像他在一個 4098×4098 的方格圖上行走，一開始他在中心，而且他到死之前都不會碰到方格圖的邊界。我們可以簡單的把地圖轉化成圖形 (Graph)，把方格當作點，而縫合怪沒有走過得邊界，轉換成邊。接下來執行 Floyd-Warshall 演算法，變可以得知任兩個方格，是不是在同一個區域，接下來套用互斥集合 (Disjoint sets) 的資料結構，便能算出有多少個分開的區域了。然而，夜色鎮太大了，而上面這個作法要處理一千六百多萬個格子，在實務上這方法不夠快。請找出更好的方法解決這個問題。

Input Format

測試輸入的第一行只有一個整數 T ($T \leq 30$) 代表有多少組測試資料。每筆測試資料只有一行，有一個描述縫合怪移動的序列，`U` 代表往畫面上方、`D` 代表下方、`L` 代表左方、`R` 代表右方，序列長度不會超過 2048。

Output Format

對每筆測試資料，輸出有多少個分開的區域。

Sample Input

```
4
LURD
LUDR
LDRDRURRDLUURURD
LLUURRDDRULLLLUR
```

Sample Output

```
2
1
2
5
```

Problem G
Kurt Gödel

Time limit: 10 seconds
Memory limit: 512 megabytes

Problem Description

> 庫爾特・哥德爾是個奧地利數學家、哲學家。他受了艾薩克・牛頓與伊曼努爾・康德等科學家哲學家的影響，也影響了其他數學家與哲學家，如伯特蘭・羅素。他運用了「哥德爾數」的概念來證明著名的「哥德爾不完備定理」，探討了數學與邏輯本身的疆界與極限，這個成就讓他得與亞里斯多德同列於歷史上最重要的邏輯學家之林。

讓我們說明一下哥德爾數的系統如何運作。首先，將所有的符號賦予一個數字。在本題當中，我們只考慮大寫英文字母 (A 到Z)，且給予這些字母，1 到 26 的編號。將A 編為 1、B 編為 2，並以此類推。舉例來說，KURT 將以 $(11, 21, 18, 20)$ 這個數列表示。接下來，將這個數列用質數的乘積來替代：將數列 $(a_1, a_2, \ldots, a_n)$ 以 $2^{a_1} \cdot 3^{a_2} \cdot \cdots \cdot p_n^{a+n}$ 表示，其中 p_i 是第 i 個質數。KURT 所對應的哥德爾數，即是：$2^{11} \cdot 3^{21} \cdot 5^{18} \cdot 7^{20} = 652074444016292692118429064843750000000000$。

對應一個長單字的哥德爾數可能非常大。你朋友阿爾博特想要用哥德爾數留一個訊息給你，但他懶得寫下所有的位數。阿爾博特用了使用三個數 (ℓ, r, p) 來表示一個哥德爾數 g，其中 ℓ 是訊息的英文字母數，p 是個質數，且 $r \equiv g \pmod{p}$。然而，阿爾博特並沒有注意到，兩個相異的字，可能可以產生出相同的 (ℓ, r, p)。舉例來說，EA 跟JA 的哥德爾數分別是 96 與 3072，但阿爾博特可以用 $(2, 3, 31)$ 來表達這兩個數字。請試著解開阿爾博特留給你的訊息吧。

Input Format

輸入資料最多有 30 組測試資料。每組測試資料只有一行。該行有三個整數。第一個是 ℓ $(1 \le \ell \le 8)$ 代表訊息的單字數量。第二個是 r、第三個是 p，要表示的哥德爾數對 p 取餘數得結果為 r，且保證 $0 \le r < p < 2^{31}$。當讀到一行有 0 代表輸入結束。

Output Format

如果 (ℓ, r, p) 所代表的英文單字是唯一的，請輸出該單字。如果能代表的英文單字超過一個，則輸出 "ambiguous"。如果完全沒有辦法代表任何英文單字，則輸出 "not a word"。

Sample Input

```
1 2 3
3 11385 13339
7 123123123 2089141547
7 0 2089141547
0 0 0
```

Sample Output

```
ambiguous
COW
ambiguous
not a word
```

Problem H
Richard Hamming

Time limit: 1 second
Memory limit: 256 megabytes

Problem Description

兩個 n 維向量 $\vec{v} = (v_1, \ldots, v_n)$、$\vec{u} = (u_1, \ldots, u_n)$ 的漢明距離 $d_H(\vec{v}, \vec{u})$ 定義為：

$$d_H(\vec{v}, \vec{u}) = |\{i : v_i \neq u_i \text{ and } i \in \{1, \ldots, n\}\}|$$

即兩個向量相異處的數量。舉例來說，$(1, 2, 3, 4, 5)$ 與 $(1, 0, 0, 4, 5)$ 的漢明距離是 2，因為僅在第二與第三個位置是相異的。請撰寫一個程式來計算兩個 n 維向量的漢明距離。

Input Format

測試輸入第一行有一個整數 T $(T \leq 100)$ 代表有多少組測試資料。每一組測試資料有三行。其中第一行有一個整數 n $(0 < n \leq 50)$ 代表向量的維度。第二行有 n 個整數 $v_1, \ldots, v_n$，第三行有 n 個整數 $u_1, \ldots, u_n$。可以假定 $v_1, \ldots, v_n, u_1, \ldots, u_n \in \{0, 1, \ldots, 99\}$。

Output Format

針對每一組測試資料，輸出 $(v_1, \ldots, v_n)$ 與 $(u_1, \ldots, u_n)$ 之間的漢明距離。

Sample Input

```
2
3
1 2 3
3 2 1
4
1 0 1 0
1 0 1 1
```

Sample Output

```
2
1
```

附註

理察・漢明是 1968 年圖靈獎得主，因數值方法、錯誤偵測碼與錯誤更正碼等領域的卓著貢獻，於 1968 年獲得圖靈獎。

Problem I
Oracle

Time limit: 3 seconds
Memory limit: 256 megabytes

Problem Description

預言者 (Oracle) 是一種計算機科學的概念：一個黑盒子 (不知道內部運作原理的東西)，能夠以單一個操作回答特定問題，主要用於分析決定型問題的計算複雜度。而預言者的起源，可以追溯至古希臘。其中最著名的，可能是德爾菲的預言者皮提亞，太陽神阿波羅的祭司。皮提亞不是一個人的名字，而是德爾菲預言者的稱號。自西元前七世紀至西元四世紀為止，希臘人都向皮提亞求預言、進行諮詢。阿波羅自己的女兒費莫奴 (Phemonoe)，可能是第一個德爾菲的預言者，是這個問題的主角。你可能會問「為什麼這問題沒有用姓 I 的計算機科學家做為主題？」嗯，命題組無法找到一個姓 I 的計算機科學家可以跟這問題扯上關係，請你認命吧。

費莫奴正在一連串 n 場的賭局中。這些賭局依序進行，不能夠回到過去的賭注下注，因為阿波羅的神力無法進行時間旅行。但費莫奴可以選擇不要參加特定賭局。每一場賭局她可以下注，並依據賭局的結果，獲得下注金額倍數的報酬或損失。一般而言，賭客並不會事前知道賭局結果是什麼，可費莫奴是德爾菲的預言者呢！阿波羅早把所有賭局的結果會是什麼了，但也警告費莫奴不要太貪心，可疑的贏了太多並不是好主意。眾神之王宙斯可能會因為阿波羅洩漏結果而感到不快。

不幸的，在幾杯上好希臘紅酒下肚後，費莫奴的判斷力變得不那好了。隨著他的信心增長，他也開始加高他的賭注。作為她的顧問，你的工作是最大化費莫奴的獲利。但要注意的是，他可能不會聽從你的意見太多，他會自己決定要參與賭局的總數，你只能建議她該從所有 n 場賭局中，選出哪些場參加。

Input Format

第一行僅有一個數字 T ($T \leq 20$) 代表有幾組測試資料。每組測試資料的第一行，會有一個正整數 n ($n \leq 50000$) 代表有多少場賭局可以參加。每組測資的第二行會有 n 個 $p_1, \ldots, p_n$ 個整數，代表每場賭局的倍率。費莫奴在參加第 i 場時，將可獲得下注金額乘上 p_i 的報酬，負數代表賠錢。你可以假定 $|p_i| \leq 50000$ 對所有的 $i \in \{1, \ldots, n\}$ 都成立。每組測試資料的第三行有兩個非負整數 a 和 b，這將描述費莫奴下注的金額如何隨著信心上升而提昇。費莫奴將會在她參與的第 j 場下注 $j^2 + aj + b$，且保證 $a^2 \geq 4b$。

Output Format

針對每筆測試資料，輸出 n 個整數，代表費莫奴參與 $1, \ldots, n$ 場賭局各自能獲得的最高收益。

Sample Input

```
2
3
1 2 3
0 0
5
1 -1 1 -1 1
3 2
```

Sample Output

```
3 14 36
6 18 38 44 26
```