Problem A
Number guessing
Input file: *pa.txt*

Problem Statement

A standard dice is given in Figure 1-(a). The opposite sides of a dice always add up to seven. Given three standard dices which are stacked as shown in Figure 1-(b) and based on the numbers shown on front view and side view, you are asked to write a program to determine what are the numbers on the top and the base surfaces of this stack of dices?

圖 1-(a) 是一個標準的骰子，骰子相背對的兩面其點數和加起來總是 7。現在給三個標準的骰子， 疊成一直列如圖 1-(b)，骰子正面及旁邊的點數如圖所示。請您寫一個程式來決定圖 1-(b)最上面骰子的上方面的點數，及最下面骰子的下方面的點數。

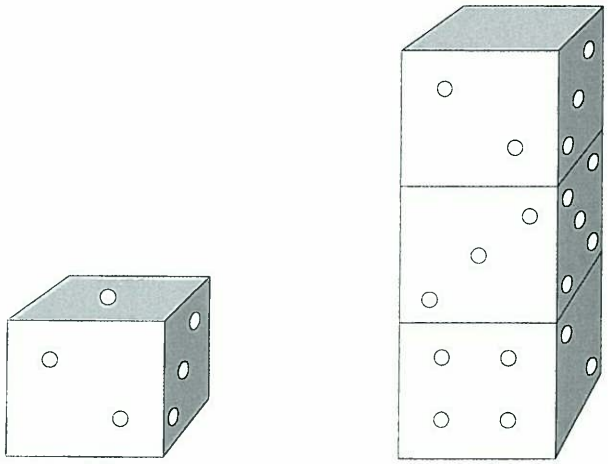


Figure 1 (a) A standard dice. (b) A stack of three standard dices.

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file. Each test case consists of 3 lines which represent a pair of numbers from top to bottom dices, and each line contains a pair of front face number and side face number.

Output Format

For each test case, print out all of the following on a single line separated by blanks:

Case *i*: The top face is ___ and the bottom face is ___.

where *i* is the test case number.

Sample Input

```
2
2 3
3 5
4 2
/
5 6
6 2
6 5
.
```

Sample Output

```
Case 1: The top face is 1 and the bottom face is 6
Case 2: The top face is 4 and the bottom face is 4
```

Problem B
Buddy Memory Management
Input file: *pb.txt*

Problem Statement

There is a computer with 1M memory. It can execute several programs at the same time. Company X decides to design a memory management system for the 1M memory, called *BUDDY*. The rules of allocating memory to a program are as follows: Suppose a program needs *m* bytes of memory and $2^L < m \leq 2^U$, where U=L+1. *BUDDY* will allocate 2^U bytes to the program. In order to allocate the 2^U bytes to the program, *BUDDY* will divide the free memory in a hierarchical and power-of-two manner.

For example, suppose a program A requests 100K bytes memory from a free 1M bytes memory. According to the design, *BUDDY* will give 128K (power of 2 that is greater than 100K) to program A. In order to give a 128K memory segment, *BUDDY* first divides the 1M free memory into two 512K memory segments. Next, *BUDDY* chooses one of lowest 512K memory segment and divides it into two 256K memory segments. The process continues until a 128K memory segment is created and allocated to program A. The final segment layout of 1M memory is like Fig.1. We say segment 1 and segment 2 are *buddies* because they are divided from the same 256K segment.

有一種電腦具有 1 Mega Byte 記憶體。此電腦可以同時執行許多程式。一家軟體公司 X 為此電腦設計了一個記憶體管理系統叫做**伙伴系統**。此伙伴系統負責配置記憶體給電腦程式。伙伴系統配置記憶體的規則如下：假設一個電腦程式需要 *m* 個 bytes 的記憶體， $2^L < m \leq 2^U$ 而且 U=L+1，則伙伴系統會配置給該程式 2^U bytes。不過，除了上述的規則，伙伴記憶體系統還必須把可用的記憶體(free memeoru)做切割。切割的方法是一個階層式的 2 的次方。

例如，假設有一個程式 A 要求 100K bytes 記憶體。根據設計，伙伴系統會給 128K 的記憶體，因為 128K 是大於 100K 的第一個數值是 2 的次方。不過，為了配置這 128K 的記憶體，伙伴系統首先把 1M 可用的記憶體空間切割成兩個 512K 的記憶體區段。接著伙伴系統挑位址較低的 512K 區段並繼續把它切割成 2 個 256K 記憶體區段。這個步驟一直持續到形成 128K 的記憶體區段，然後配置給程式 A。最後 1M 記憶體的配置圖如圖一。在圖一中，我們說記憶體區段 1 與區段 2 是**伙伴**，因為他們都是從同一個 256K 區段被切割出來的。

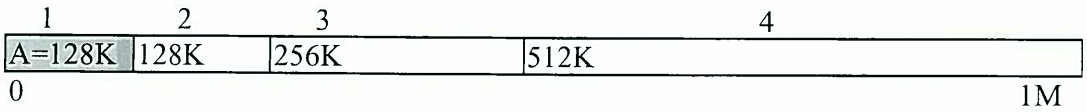


Figure 1

Suppose now another program B requests 60K memory. The design searches among the free memory segments and looks for the smallest free segment that is larger than 60K. By the rule, segment 2 is chosen for program B. However, according to the design, 128K is too large. We need to divide the segment into two 64K segments like Fig.2 and allocate the first 64K segment to program B. In Fig. 2, we say segment 2 and segment 3 are *buddies* because they are divided the same segment. On

the other hand, segment 1 and 2 are not buddies, although they are adjacent segments.

假設現在有另外一個程式 B 需要 60K 記憶體。伙伴系統的設計會先去搜尋可用的記憶體區段。從可用的記憶體區段中找出一個大於 60K 但是是最小的區段。根據這個規則，記憶體區段 2 會被選來配置給程式 B。不過，根據伙伴系統的規則，128K 太大。伙伴系統會把此區段分割成兩個 64K 的可用記憶體區段。然後把低位址的區段配置給程式 B，如圖二。在圖二中，我們說區段 2 與區段 3 是伙伴，因為它們都是從一個 128K 的區段中分割出來的。不過區段 1 與區段 2 已經不是伙伴了，即使它們是相鄰的。

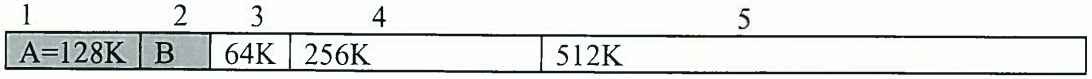


Figure 2

Let's continue with the example. Suppose a program C requests 200K and the layout becomes Fig. 3.

讓我們繼續這個例子。假設有另外一個程式 C 要求 200K，則記憶體配置圖如圖三。

Figure 3

After C is loaded, suppose program B releases its memory segment. The memory layout will become Fig. 4, where the two 64K segments are merged into a 128K free segments because they are buddies.

假設在程式 C 被載入後，程式 B 釋放他所佔的記憶體。則記憶體配置圖會變成圖四。其中兩個 64K 的可用記憶體區段會結合成一個 128K 的可用記憶體區段因為它們是伙伴。

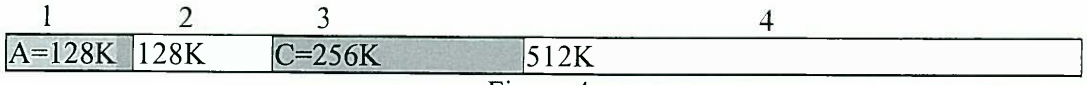


Figure 4

Now, please write a program to implement such a design of BUDDY.

請寫一個程式實做上述的伙伴系統設計。

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file.

For each test case, it contains a sequence of program requests. Each program request is described by (*m s*), where *m* is a string, the name of a program and *s* is the requested size (in Kbytes) of memory. When *s* = *-1,* it means the program requests to release its memory. When request is (@ -1), it means the end of a test case. **Note that each test case must begin with 1M free memory.**

輸入檔的第一列是一個整數 N，1≦ N ≦ 10, 表示測試的案例數。兩個測試的案例中間有一個 "/" 作為分隔。測試檔案的最後一列有一個 "." 表示測試檔案結束。

每一個測試案例，有一連串的記憶體需求。每個需求表示成 (*m s*) 其中 *m* 是一個字串，為程式的名稱。*s* 是該程式要求的記憶體大小 (以 Kbytes) 為單位。當 s = -1 時，表示該程式想要釋放他所佔用的記憶體。當需求為 (@ -1)時，表示一個測試案例結束。**請注意，每個測試案例都從 1M 的可用記憶體空間執行起。**

Output Format

At the end of each test case, please output the memory layout in one line. The layout begins with the lowest memory segment. Each segment is output by "(*m s*)". For free segment, please use "free" as the name of the segment.

在每一個測試案例結束，請輸出此時記憶體的配置圖。輸出時請從低位址開始。每個記憶體區段輸出為(m s)。當一個記憶體區段為可用記憶體時，請用"free"作為程式名稱。

Sample Input

```
2
A 100
B 60
C 200
B -1
@ -1
/
A 100
B 500
C 200
A -1
C -1
@ -1
.
```

Sample Output

```
(A 128) (free 128) (C 256) (free 512)
(free 512) (B 512)
```

Problem C
Secrete Code
Input file: *pc.txt*

Problem Statement

The Department of Defense (DoD) has developed a simple scheme for transmitting secrete messages across the network. The message is first placed in a binary tree, one character per node, so that the post-order traversal of the binary tree results in the message itself. When transmitting the message, the strings that resulted from the pre-order traversal and the in-order traversal of that binary tree are sent across the network. When the two strings are received, the original message can be reconstructed using those two strings along. Please write a program to decode the message from the two received strings.

The message contains only distinct alphanumeric characters {A-Z, a-z, 0-9}; and the message length is at most 60 characters long. Upper- and lower-case letters are considered distinct ASCII characters.

國防部發展了一個簡單的編碼方式來傳遞訊息。編碼方式如下：首先將要傳遞的訊息放在一個二元樹內，每一個節點放一個字元，且後序巡行（Post-order traversal）所產生的字串即是解碼後的訊息。當傳遞此訊息時，我們將這個二元樹的前序巡行及中序巡行所得的字串傳送出，當對方接收到這兩個字串，則可將其轉換成原先存在二元樹中的訊息，請寫一程式來做這件事情。

我們要傳送的訊息只包含不同的大小寫英文字母及０到９的數字字元，同時該訊息長度最多爲６０個字元，大小寫英文字元視爲不同。

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file.

For each of the test cases, there are two lines of input. The first line contains the coded message using pre-order traversal. The second line contains the coded message using in-order travel.

Output Format

For each test case, the output contains a line with the size of the minimum partition as described above.

Sample Input

```
2
abc*e
cb*ae
/
A_Dce
D_Ace
.
```

Sample Output

```
c*bea
D_ecA
```

Problem D
Longest Common Circular Subsequence
Input file: *pd.txt*

Problem Statement

For any string X, we use $|X|$ to denote the length of X. For any two strings X and Y, we say that Y is a *circular subsequence* of X if there are indices $i_1, i_2, \ldots, i_k, i_{k+1}, \ldots, i_{|Y|}$, such that (1) $1 \le i_{k+1} < i_{k+2} < \ldots < i_{|Y|} < i_1 < i_2 < \ldots < i_k \le |X|$, and (2) $X[i_j] = Y[j]$ holds for each index $j = 1, 2, \ldots, |Y|$. For example, "abcdef" is circular subsequence of "xcxdxxexxfaxbx".

Given two strings A and B, you are asked to compute a longest string C such that C is a circular subsequence of both A and B. If there are more than one such C, please output the one with the least alphabetical order.
You may assume that (i) the input strings consist of only those 26 lower-case English characters, and (ii) the length of S is at most 100.

對任何字串 X, 我們用$|X|$ 來代表 X 的長度. 對任何兩個字串 X 與 Y, 如果下列條件成立,則我們稱 Y 是 X 的一個 *circular subsequence* of X

我們可以找得到索引 $i_1, i_2, \ldots, i_k, i_{k+1}, \ldots, i_{|Y|}$ 以致於

(1) $1 \le i_{k+1} < i_{k+2} < \ldots < i_{|Y|} < i_1 < i_2 < \ldots < i_k \le |X|$, 而且

(2) $X[i_j] = Y[j]$ 對任何 $j = 1, 2, \ldots, |Y|$的索引均成立.

例如, "abcdef" 是 "xcxdxxexxfaxbx"的一個 is circular subsequence.

給定 A 與 B 兩個字串, 本題要求找出字串 C 使得 C 是 A 與 B 最長的共同 circular subsequence. 如果有好幾個最長的共同 circular subsequence,則請輸出按照字典順序排列中最小的一個.

你可以假設(i) 所輸入的字串都是由 26 的英文小寫字母構成, 以及(ii) 輸入的字串長度均不超過 100.

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file. For each of the test cases, there are two lines of input strings.

輸入檔的第一行為一個整數 N($1 \le N \le 10$) 代表本題的測試資料筆數.任兩筆測試資料之間用一行'/' 字元隔開. 測試資料的最後一行用一個'.'字元代表檔案結束. 每筆測試資料則由兩行英文字串構成,分別代表該筆測試資料的兩個輸入字串 A 與 B.

Output Format

For each of the test cases, print on one line the longest common circular subsequence with the least lexicographical order.

對每筆測試資料組,請印出最長的 common circular subsequence. 如果有好幾個可能的答案,則請輸出按照字典順序排列中最小的一個(the least lexicographical order)。

Sample Input

```
2
xcxdxxexxfaxbx
bycyydyeyyfyayyy
/
jxwxhxuxixvx
zizvzjzwzzuh
.
```

Sample Output

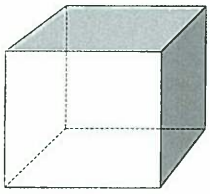
```
abcdef
hij
```

Problem E
Cube mapping
Input file: *pe.txt*

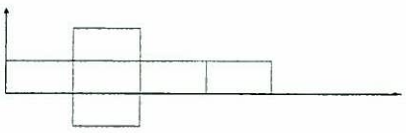
Problem Statement

If seven of the twelve edges of hollow cube are cut as shown in Figure 1-(a) and the faces then opened out, the result would be a cross shape as shown in Figure 1-(b). However cutting different choices will produce different shapes. Now by given the vertices of six faces of an opened out shape. You are asked to write a program to determine whether they can form a cube or not.

圖 1-(a) 是一個空心立方體，包含了 6 個面及 12 個邊，若把其中七個邊剪開，然後展開可以得到如圖 1-(b) 的展開圖。但是如果剪開的邊不一樣，得到的展開圖也會不一樣。現在給一種展開圖，六個面的四個頂點的座標，請您寫一個程式來決定這六個面是否能摺回一個空心立方體。



(a)



(b)

Figure 1 (a) A hollow cube (b) The faces after opened out.

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the

number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file. Each test case consists of six lines, and each line contains four pairs of vertex coordinates (x1, y1) (x2, y2) (x3, y3) (x4, y4).

Output Format

For each test case, print out all of the following on a single line:

```
Case i: Yes, it forms a cube          or
Case i: No, it cannot form a cube
```

Sample Input

```
2
0 0 1 0 1 1 0 1
1 0 2 0 2 1 1 1
1 0 1 -1 2 -1 2 0
1 1 2 1 2 2 1 2
2 0 3 0 3 1 2 1
3 0 4 0 4 1 3 1
/
0 0 0 1 1 1 1 0
1 0 1 1 2 1 2 0
2 0 2 1 3 1 3 0
2 1 2 2 3 2 3 1
3 0 3 1 4 1 4 -1
2 -1 2 0 3 0 3 -1
.
```

Sample Output

```
Case 1: Yes, it forms a cube
Case 2: No, it cannot form a cube
```

Problem F
Minimum Integer Combination
Input file: *pf.txt*

Problem Statement

Given a sequence of positive integers V[1],...,V[N] and a positive integer X, we say X can be generated with V[i]'s if there exist integers C[1],...,C[N] such that X = C[1]*V[1] +...+ C[N]*V[N]. The combination may not be unique. There is a fact : X can be generated with V[1],...,V[N] if and only if X is a multiple of the greatest common divisor of V[i]'s. Among all the possible combinations, we are interested in the one with |C[1]| +...+|C[N]| minimized (i.e., minimize the sum of absolute values of C[i]'s). For example, let X= 4 and V[1]=3, V[2]=5. Then we can represent 4 as -2*3 + 2*5, which has |-2|+|2| = 4 and achieves the minimum.

In this problem, given N, X and V[i]'s, you are asked to write a program to calculate the smallest possible |C[1]| +...+|C[N]|.

給予一正整數列 V[1]，...，V[N]及一正整數 X，如果存在一整數列 C[1]，...，C[N]使得 x=C[1]*V[1]+...+C[N]*V[N]，則我們說 X 可以由上述 V[i]產生。但上述的組合並非唯一。提示：X 可以由 V[1]，...，V[N]產生，若且唯若 X 是所有 V[i] 之 GCD 的倍數。在所有的可能組合中，我們想要找出一組合，使得 |C[1]|+...+|C[N]|的絕對值最小，例如，設 X=4 且 V[1]=3，V[2]=5，則依前述定義-2*3+2*5=4，使得|-2|+|2|=4 的值最小。

在本題中，給予 N，X，及數列 V[i]，試寫一程式求出|C[1]|+...+|C[N]|的最小值。

Input File Format

The first line of the input file contains an integer N, $1 \le N \le 10$, indicating the number of test cases. There is a single line containing a '/' character separating two consecutive test cases. The last line of the input file contains a '.' character denoting the end of the input file. For each of the test cases, there is one line that shows the values of N, X and V[1], ..., V[N], where N ≤ 200, X ≤ 10000 and each V[i] ≤ 10000. Note that in the file we use space to separate the numbers. The last line of the input file contains a '.' character denoting the end of the input file.

Output Format

For each test input, the output contains a line with the minimum sum as described above.

Sample Input

```
3
2 2 2 4
/
2 4 3 5
/
5 7 1 3 4 5 8
.
```

Sample Output

```
1
4
2
```