4^o Treino para Alunos da UFPR

8 de Fevereiro de 2013

Sevidor BOCA:
`http://maratona.c3sl.ufpr.br/boca/`

Organizadores:
Vinicius Kwiecien Ruoso, Ricardo Tavares de Oliveira e Flávio Zavan

Lembretes:

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos.
- Utilize o *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos.

A: Alice Sieve

`File: alicesie.[c|cpp|java|pas]`

Alice has recently learned to use the Sieve of Eratosthenes, an ancient algorithm for finding all prime numbers up to any given limit. As expected, she was really impressed by it's simplicity and elegancy.

Now, she has decided to design her own sieve method: The Sieve of Alice, formally defined by the following procedure, which determines the Sieve of Alice up to a given limit N.

1. Create a list of consecutive integers from N to 2 (N, $N-1$, $N-2$, ..., 3, 2). All of those $N-1$ numbers are initially unmarked.
2. Initially, let P equal N, and leave this number unmarked.
3. Mark all the proper divisors of P (i.e. P remains unmarked).
4. Find the largest unmarked number from 2 to $P-1$, and now let P equal this number.
5. If there were no more unmarked numbers in the list, stop. Otherwise, repeat from step 3. Unfortunately, Alice has not found an useful application for it's Sieve. But she still wants to know, for a given limit N, how many integers will remain unmarked.

Input

The first line contains an integer T, the number of test cases ($1 \leq T \leq 10^4$). Each of the next T lines contains an integer N ($2 \leq N \leq 10^6$).

Output

Output T lines, one for each test case, containing the required answer.

Sample Input	**Sample Output**
3 2 3 5	1 2 3

B: Beautiful Strings

`File: beautiful.[c|cpp|java|pas]`

When John was a little kid he didn't have much to do. There was no internet, no Facebook, and no programs to hack on. So he did the only thing he could... he evaluated the beauty of strings in a quest to discover the most beautiful string in the world.

Given a string s, little Johnny defined the beauty of the string as the sum of the beauty of the letters in it.

The beauty of each letter is an integer between 1 and 26, inclusive, and no two letters have the same beauty. Johnny doesn't care about whether letters are uppercase or lowercase, so that doesn't affect the beauty of a letter. (Uppercase 'F' is exactly as beautiful as lowercase 'f', for example.)

You're a student writing a report on the youth of this famous hacker. You found the string that Johnny considered most beautiful. What is the maximum possible beauty of this string?

Input

The input file consists of a single integer m followed by m lines.

Output

Your output should consist of, for each test case, a line containing the string "Case #x: y"where x is the case number (with 1 being the first case in the input file, 2 being the second, etc.) and y is the maximum beauty for that test case.

Constraints

$5 \leq m \leq 50$

$2 \leq$ length of $s \leq 500$

Sample Input	**Sample Output**
5 ABbCcc Good luck in the Facebook Hacker Cup this year! Ignore punctuation, please :) Sometimes test cases are hard to make up. So I just go consult Professor Dalves	Case #1: 152 Case #2: 754 Case #3: 491 Case #4: 729 Case #5: 646

C: Irmãos

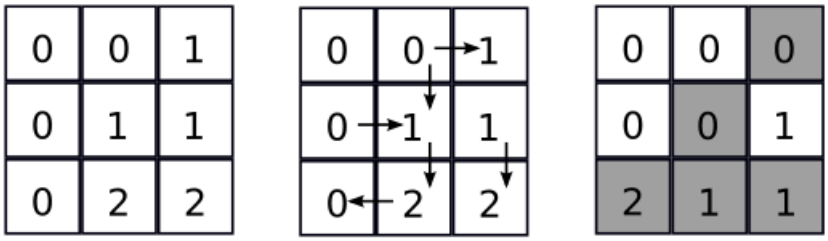
`File: irmaos.[c|cpp|java|pas]`

O reino de ACM era comandado por um grande Rei que era obcecado por ordem. O reino tinha forma retangular, e o rei dividiu o território em uma grade de pequenos países retangulares. Antes de morrer, o Rei distribuiu os países entre seus filhos.

No entanto, ele não sabe que seus filhos desenvolveram uma estranha rivalidade; o primeiro herdeiro odiava o segundo herdeiro, mas não os demais; o segundo herdeiro odiava o terceiro herdeiro, mas não os demais, e assim por diante. Finalmente, o último herdeiro odiava o primeiro herdeiro, mas não os demais.

Logo que o Rei morreu, essa estranha rivalidade entre os filhos do Rei desencadeou uma guerra generalizada no reino. Os ataques ocorriam apenas entre pares de países adjacentes (que compartilham uma borda vertical ou horizontal). Um país X atacava um país adjacente Y sempre que o proprietário de X odiava o proprietário de Y. O país atacado era sempre conquistado pelo irmão atacante. Por uma questão de honra, todos os ataques aconteciam simultaneamente, e um conjunto de ataques simultâneos recebia o nome de batalha. Após um certo número de batalhas, os filhos sobreviventes assinaram um tratado de trégua e nunca mais batalharam.

Por exemplo, se o Rei tinha três filhos (0, 1 e 2) a figura abaixo mostra o que acontece na primeira batalha dada uma distribuição inicial das terras:



Terras antes da batalha. | Ataques durante a batalha. | Terras após a batalha. Alterações em cinza.

Você foi contratado para ajudar um historiador da ACM a determinar, dados o número de herdeiros, a distribuição inicial de territórios e o número de batalhas, qual a distribuição final de territórios após as batalhas.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém quatro inteiros N, R, C e K separados por um espaço em branco. N é o número de herdeiros ($2 \leq N \leq 100$), R e C são as dimensões do reino ($2 \leq R, C \leq 100$) e K é o número de batalhas ($1 \leq K \leq 100$). Herdeiros são identificados por inteiros de 0 até N-1. Cada uma das seguintes R linhas contém C inteiros $H_{r,c}$ separados por um espaço em branco, representando a distribuição inicial de terras: $H_{r,c}$ é o proprietário inicial do país na linha r e coluna c ($0 \leq H_{r,c} \leq N-1$).

O último caso de teste é seguido de uma linha contendo quatro zeros separados por um espaço em branco.

Saída

Para cada caso de teste, seu programa deve imprimir R linhas com C inteiros cada, separados por um espaço em branco, no mesmo formato da entrada, representando a distrubuição das terras após todas as batalhas.

Exemplo de entrada	**Exemplo de saída**
3 4 4 3 0 1 2 0 1 0 2 0 0 1 2 0 0 1 2 2 4 2 3 4 1 0 3 2 1 2 8 4 2 1 0 7 1 6 2 5 3 4 0 0 0 0	2 2 2 0 2 1 0 1 2 2 2 0 0 2 0 0 1 0 3 2 1 2 7 6 0 5 1 4 2 3

D: Beautiful Matrix

`File: matrix.[c|cpp|java|pas]`

You've got a 5×5 matrix, consisting of 24 zeroes and a single number one. Let's index the matrix rows by numbers from 1 to 5 from top to bottom, let's index the matrix columns by numbers from 1 to 5 from left to right. In one move, you are allowed to apply one of the two following transformations to the matrix:

- Swap two neighboring matrix rows, that is, rows with indexes i and $i + 1$ for some integer $i(1 \leq i < 5)$;
- Swap two neighboring matrix columns, that is, columns with indexes j and $j + 1$ for some integer $j(1 \leq j < 5)$.

You think that a matrix looks *beautiful* if the single number one of the matrix is located in its middle (in the cell that is on the intersection of the third row and the third column). Count the minimum number of moves needed to make the matrix beautiful.

Input

The input starts with T, the number of test cases. Then, each test case is given.

A test case consists of five lines, each line contains five integers: the $j-th$ integer in the $i-th$ line of the case represents the element of the matrix that is located on the intersection of the $i-th$ row and the $j-th$ column. It is guaranteed that the matrix consists of 24 zeroes and a single number one.

Output

For each test case, print a single integer - the minimum number of moves needed to make the matrix beautiful.

Sample Input	**Sample Output**
2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	3 1

E: Adding Reversed Numbers

`File: reversed.[c|cpp|java|pas]`

The Antique Comedians of Malidinesia prefer comedies to tragedies. Unfortunately, most of the ancient plays are tragedies. Therefore the dramatic advisor of ACM has decided to transfigure some tragedies into comedies. Obviously, this work is very hard because the basic sense of the play must be kept intact, although all the things change to their opposites. For example the numbers: if any number appears in the tragedy, it must be converted to its reversed form before being accepted into the comedy play.

Reversed number is a number written in arabic numerals but the order of digits is reversed. The first digit becomes last and vice versa. For example, if the main hero had 1245 strawberries in the tragedy, he has 5421 of them now. Note that all the leading zeros are omitted. That means if the number ends with a zero, the zero is lost by reversing (e.g. 1200 gives 21). Also note that the reversed number never has any trailing zeros.

ACM needs to calculate with reversed numbers. Your task is to add two reversed numbers and output their reversed sum. Of course, the result is not unique because any particular number is a reversed form of several numbers (e.g. 21 could be 12, 120 or 1200 before reversing). Thus we must assume that no zeros were lost by reversing (e.g. assume that the original number was 12).

Input

The input consists of N cases. The first line of the input contains only the positive integer N. Then follow the cases. Each case consists of exactly one line with two positive integers separated by space, both not greater than 10^5. These are the reversed numbers you are to add.

Output

For each case, print exactly one line containing only one integer - the reversed sum of two reversed numbers. Omit any leading zeros in the output.

Sample Input	**Sample Output**
3 24 1 4358 754 305 794	34 1998 1