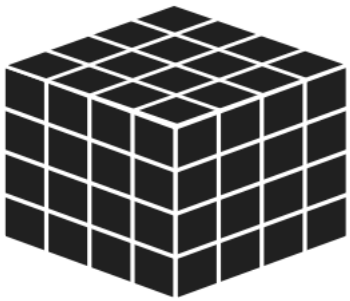
Cubo Preto

Nome do arquivo: `cubo.c`, `cubo.cpp`, `cubo.java`, `cubo.js` *ou* `cubo.py`

Ana comprou um cubo de madeira de lado N cm (ou seja, dimensões $N \times N \times N$ centímetros) e o pintou todo de preto. Depois disso, ela cortou o cubo em N^3 cubinhos de lado 1 cm (ou seja, dimensões $1 \times 1 \times 1$ centímetro). Após o corte, alguns cubinhos terão nenhuma face pintada de preto, alguns terão exatamente uma face pintada, alguns terão exatamente duas faces pintadas e outros terão exatamente três faces pintadas.

Abaixo podemos ver um cubo de lado 4 cm ($N = 4$) após Ana pintá-lo e cortá-lo.



Ana contou quantas faces estavam pintadas em cada cubinho cortado do cubo acima e concluiu que, entre os 64 cubinhos, existem 8 cubinhos com nenhuma face pintada de preto, 24 cubinhos com exatamente uma face pintada, 24 cubinhos com exatamente duas faces pintadas e 8 cubinhos com exatamente três faces pintadas.

A sua tarefa é: dada a dimensão N do lado do cubo em centímetros, determine quantos cubinhos terão exatamente nenhuma, uma, duas e três faces pintadas de preto após Ana pintar e cortar o cubo.

Entrada

A entrada contém uma única linha com um único inteiro N, a dimensão do cubo em centímetros.

Saída

Seu programa deverá imprimir quatro linhas, cada uma contendo um único inteiro:

- A primeira linha deve conter o número de cubinhos com nenhuma face pintada de preto.
- A segunda linha deve conter o número de cubinhos com exatamente uma face pintada.
- A terceira linha deve conter o número de cubinhos com exatamente duas faces pintadas.
- A quarta e última linha deve conter o número de cubinhos com exatamente três faces pintadas.

Restrições

- $2 \leq N \leq 100$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (30 pontos):** $N = 5$.
- **Subtarefa 3 (70 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4	8 24 24 8

Exemplo de entrada 2	**Exemplo de saída 2**
2	0 0 0 8

Alfabeto Alienígena

Nome do arquivo: `alfabeto.c`, `alfabeto.cpp`, `alfabeto.java`, `alfabeto.js` *ou* `alfabeto.py`

Mais uma vez, o OBI (Órgão Brasileiro de Inteligência) está preocupado com a possibilidade da existência de vida alienígena. Os diretores do órgão suspeitam que os alienígenas existem, conseguiram se infiltrar dentro da instituição e tem se comunicado secretamente. Os agentes do OBI se comunicam usando o dispositivo de mensagens oficial do órgão, que possui as seguintes teclas: letras maiúsculas de `A` a `Z`, letras minúsculas de `a` a `z`, dígitos de `0` a `9`, operadores aritméticos (`+`, `-`, `*`, `/`), *hashtag* (`#`) e ponto de exclamação (`!`).

O OBI descobriu que, sempre que dois alienígenas se comunicam entre si usando o dispositivo, eles usam um alfabeto alienígena que possui um conjunto específico de símbolos. Assim, uma mensagem pode ter sido escrita por alienígenas se, e somente se, todos os símbolos que compõem ela pertencem ao alfabeto alienígena. Por exemplo, se o alfabeto alienígena for composto pelas caracteres `!`, `1`, `o` e `b`, a mensagem `ob1!!` é uma mensagem que poderia ser escrita por alienígenas. Por outro lado, a mensagem `Obi!` não poderia ter sido escrita por alienígenas pois tanto o primeiro caractere `O` (maiúsculo) quanto o terceiro caractere `i` não fazem parte do alfabeto alienígena.

Você foi contratado para ajudar o OBI a identificar os invasores: dadas a lista de caracteres usados no alfabeto alienígena e uma mensagem enviada pelo dispositivo, determine se a mensagem poderia ter sido escrita por alienígenas ou não.

Entrada

A primeira linha de entrada contém dois inteiros K e N separados por um espaço em branco, indicando, respectivamente, o número de caracteres presentes no alfabeto alienígena e o número de caracteres da mensagem enviada.

A segunda linha de entrada contém K caracteres distintos representando os caracteres pertencentes ao alfabeto alienígena.

A terceira linha de entrada contém N caracteres (não necessariamente distintos) representando a mensagem enviada.

Saída

Seu programa deverá imprimir uma única linha contendo um único caractere: se a mensagem pode ter sido escrita no alfabeto alienígena, imprima a letra 'S' maiúscula; caso contrário, imprima a letra 'N' maiúscula.

Restrições

- $1 \leq K \leq 68$
- $1 \leq N \leq 1000$
- Todos os caracteres usados no alfabeto ou na mensagem pertencem à lista a seguir:

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-*/#!
```

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (33 pontos):** $K = 1$, ou seja, o alfabeto alienígena possui apenas um símbolo *(veja o exemplo 2)*.
- **Subtarefa 3 (29 pontos):** $K = 26$ e o alfabeto alienígena é exatamente o nosso alfabeto de letras minúsculas, ou seja, `abcdefghijklmnopqrstuvwxyz` *(veja o exemplo 3)*.
- **Subtarefa 4 (38 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**

`4 5`
`!1ob`
`ob1!!`
	`S`

Exemplo de entrada 2	**Exemplo de saída 2**

`1 5`
`a`
`aabab`
	`N`

Exemplo de entrada 3	**Exemplo de saída 3**

`26 32`
`abcdefghijklmnopqrstuvwxyz`
`olimpiadabrasileiradeinformatica`
	`S`

Exemplo de entrada 4	**Exemplo de saída 4**

`11 7`
`0123+-!ABCD`
`OBI!OBI`
	`N`

Dança de Formatura

Nome do arquivo: `dancs.c`, `dancs.cpp`, `dancs.java`, `dancs.js` *ou* `dancs.py`

A escola de educação básica do seu bairro está organizando uma festa de formatura para os graduandos deste ano. Para isso, eles pediram que a OBI (Organização de Brincadeiras Infantis) desenvolva uma dança que os alunos possam apresentar aos pais durante a formatura.

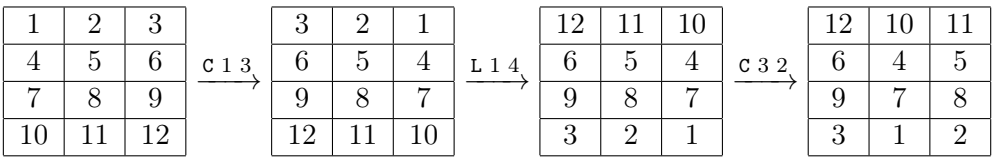
A dança da OBI é dançada em uma pista quadriculada com N linhas e M colunas, sempre com exatamente um aluno em cada quadrado do pista. Os alunos são numerados de 1 a $N \times M$ de acordo com a sua posição inicial na pista em ordem crescente de linha e coluna, nesta ordem, a partir do quadrado $(1, 1)$. O exemplo abaixo, para $N = 4$ e $M = 3$, indica o número do aluno em cada quadrado da pista no início da dança; o aluno de número 7, por exemplo, inicia no quadrado $(3, 1)$.

	Col. 1	Col. 2	Col. 3
Linha 1	1	2	3
Linha 2	4	5	6
Linha 3	7	8	9
Linha 4	10	11	12

A cada passo da dança, o professor dá aos alunos uma das duas ordens abaixo:

- "`L` a b" (onde a e b são inteiros distintos), ordenando que os alunos da a-ésima linha troquem de linha com os alunos da b-ésima linha, mantendo a coluna de cada um – ou seja, o aluno na célula $(a, 1)$ troca com o aluno na célula $(b, 1)$, $(a, 2)$ troca com $(b, 2)$ e assim por diante.
- "`C` a b" (onde a e b são inteiros distintos), ordenando que os alunos da a-ésima coluna troquem de coluna com os alunos da b-ésima coluna, mantendo a linha de cada um – ou seja, o aluno na célula $(1, a)$ troca com o aluno na célula $(1, b)$, $(2, a)$ troca com $(2, b)$ e assim por diante.

A figura abaixo ilustra o progresso da dança para $N = 4$ e $M = 3$ com os três primeiros passos sendo "`C` 1 3", "`L` 1 4" e "`C` 3 2", nesta ordem.



A escola gostou muito da dança inventada pela OBI e deseja usá-la na formatura. Porém, os pais não querem perder seus filhos de vista e pediram sua ajuda para saber quais serão as posições de seus filhos ao término da dança.

Sua tarefa é: dadas as dimensões N e M da pista de dança, a quantidade P de passos da dança e a ordem dada pelo professor a cada passo, determine qual aluno estará em cada quadrado da pista ao fim da dança.

Entrada

A primeira linha da entrada é composta por três inteiros N, M e P indicando, respectivamente, o número de linhas da pista de dança, o número de colunas da pista de dança, e o número de passos da dança.

As próximas P linhas descrevem as ordens dadas pelo professor. A i-ésima dessas linhas contém uma letra **maiúscula** O_i, que pode ser `'L'` ou `'C'`, seguida de dois inteiros distintos A_i e B_i.

- Se $O_i =$ `'L'`, o professor ordenou a troca das linhas A_i e B_i.
- Se $O_i =$ `'C'`, o professor ordenou a troca das colunas A_i e B_i.

Saída

Seu programa deverá imprimir N linhas, cada uma contendo M inteiros. O j-ésimo inteiro da i-ésima linha deve ser o número do aluno que terminará a dança na i-ésima linha e j-ésima coluna da pista.

Restrições

- $1 \leq N \leq 1\,000\,000$
- $1 \leq M \leq 1\,000\,000$
- $1 < N \times M \leq 1\,000\,000$
- $1 \leq P \leq 500\,000$
- $O_i =$ `'L'` ou $O_i =$ `'C'`
- Se $O_i =$ `'L'`, $1 \leq A_i, B_i \leq N$
- Se $O_i =$ `'C'`, $1 \leq A_i, B_i \leq M$
- $A_i \neq B_i$

Atenção: *Observe que não é possível declarar* $1\,000\,000 \times 1\,000\,000$ *inteiros (com matriz, vetor etc.) sem estourar o limite de memória (isto causaria erros no programa pois tentaria usar milhares de GB de memória). Preste atenção ao limite* $N \times M \leq 1\,000\,000$*, que garante que a pista de dança sempre terá no máximo* $1\,000\,000$ *alunos.*

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (20 pontos):**
  - $N = 1$
  - $M \leq 1000$
  - $P \leq 1000$
- **Subtarefa 3 (20 pontos):**
  - $N \leq 1000$
  - $M \leq 1000$
  - $P \leq 1000$

- **Subtarefa 4 (31 pontos):**
  - $M = 2$ *(Veja o exemplo 3.)*
- **Subtarefa 5 (29 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 3 3 C 1 3 L 1 4 C 3 2	12 10 11 6 4 5 9 7 8 3 1 2

Exemplo de entrada 2	**Exemplo de saída 2**
1 6 4 C 2 5 C 1 2 C 4 3 C 1 2	1 5 4 3 2 6

Exemplo de entrada 3	**Exemplo de saída 3**
5 2 6 C 1 2 L 1 3 L 1 4 C 2 1 L 5 3 C 2 1	8 7 4 3 10 9 6 5 2 1

Jogo do Poder

Nome do arquivo: `poder.c`, `poder.cpp`, `poder.java`, `poder.js` *ou* `poder.py`

Jonathan está empolgado com a nova sensação do momento: o *Jogo do Poder*. Este jogo é jogado em uma matriz de N linhas e M colunas, na qual cada célula possui um monstro. O monstro na linha i e coluna j possui poder $P_{i,j}$.

No início do jogo, Jonathan escolhe um dos $N \times M$ monstros para jogar. O monstro escolhido por Jonathan se torna o *herói* do jogo e começa o jogo com o poder indicado em sua célula. Jonathan pode mover o herói ortogonalmente (isto é, para cima, baixo, direita ou esquerda) na matriz enquanto o herói estiver vivo. O herói não pode sair da matriz, mas pode visitar a mesma célula múltiplas vezes.

Toda vez que o herói entra em uma célula com um monstro vivo, ocorre uma batalha entre o herói e o monstro da célula. O herói ganha a batalha se, e somente se, o seu poder for maior ou igual ao poder do monstro. Caso contrário, o herói morre e perde o jogo em *game over*. Toda vez que o herói ganha uma batalha, o monstro derrotado morre (ou seja, a célula não possui mais nenhum monstro) e, como recompensa, o poder do monstro é somado ao poder do herói (ou seja, se o herói matar o monstro da célula (i, j), o poder do herói aumenta em $P_{i,j}$).

Jonathan percebeu que o jogo pode ser injusto: mesmo que ele jogue de maneira ótima, dependendo de sua escolha de herói, pode ser possível matar todos os monstros, apenas alguns ou até mesmo nenhum monstro.

Decidido a "platinar" o jogo, Jonathan precisa saber o poder máximo que cada herói consegue alcançar (ou seja, o poder máximo possível de ser atingido ao iniciar o jogo em cada célula da matriz) se o jogo for jogado de forma ótima. Felizmente, ele descobriu que os alunos da OBI (Organização dos Bons Informáticos) recentemente resolveram o *Jogo da Vida*, seu terceiro jogo favorito (atrás do *Jogo do Poder* e do *Jogo de Corrida*, claro), então ele pediu a sua ajuda novamente! Determine, para cada herói, o poder máximo que ele consegue alcançar caso Jonathan jogue de forma ótima.

Entrada

A primeira linha de entrada contém dois inteiros N e M, o número de linhas e o número de colunas da matriz, respectivamente.

As próximas N linhas contém M inteiros cada. O j-ésimo inteiro da i-ésima linha contém o poder $P_{i,j}$ do monstro na i-ésima linha e j-ésima coluna.

Saída

O seu programa deverá imprimir N linhas, cada uma contendo M inteiros. O j-ésimo inteiro da i-ésima linha deve ser o poder máximo que Jonathan consegue alcançar caso ele escolha como herói o monstro da célula (i, j) e jogue de maneira ótima.

Restrições

- $1 \leq N \leq 100\ 000$
- $1 \leq M \leq 100\ 000$
- $1 \leq N \times M \leq 100\ 000$
- $1 \leq P_{i,j} \leq 1\ 000\ 000\ 000$ para todo $1 \leq i \leq N$ e $1 \leq j \leq M$

Atenção: *Observe que não é possível declarar* $100\,000 \times 100\,000$ *inteiros (com matriz, vetor etc.) sem estourar o limite de memória (isto causaria erros no programa pois tentaria usar dezenas de GB de memória). Preste atenção ao limite* $N \times M \leq 100\,000$*, que garante que a matriz sempre terá no máximo* $100\,000$ *células.*

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (11 pontos):**
  - $N = 1$
  - $M \leq 1000$
  - $1 \leq P_{1,j} \leq 2$ para todo $1 \leq j \leq M$
- **Subtarefa 3 (13 pontos):**
  - $N = 1$
  - $M \leq 1000$
- **Subtarefa 4 (29 pontos):**
  - $N = 1$
- **Subtarefa 5 (17 pontos):**
  - $N \leq 30$
  - $M \leq 30$
- **Subtarefa 6 (30 pontos):** Sem restrições adicionais.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima (elas não precisam ser resolvidas em ordem). Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 3 2 3 9 1 7 200	6 6 22 1 22 222

Exemplo de entrada 2	**Exemplo de saída 2**
1 7 6 3 10 1 20 7 7	9 3 54 1 54 14 14

Exemplo de entrada 3	**Exemplo de saída 3**
5 6 10 10 10 10 10 10 10 10 1 1 1 10 10 10 10 1 10 10 10 10 10 4 10 10 10 10 10 10 10 2	250 250 250 250 250 250 250 250 8 8 8 250 250 250 250 8 250 250 250 250 250 8 250 250 250 250 250 250 250 2