Competidor(a): ____

Número de inscrição: ________________–______ *(opcional)*

OBI2018

Caderno de Tarefas

Modalidade **Programação** • **Nível 2** • Fase **Nacional**

25 de agosto de 2018

A PROVA TEM DURAÇÃO DE **5** HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

v1.0

Instruções

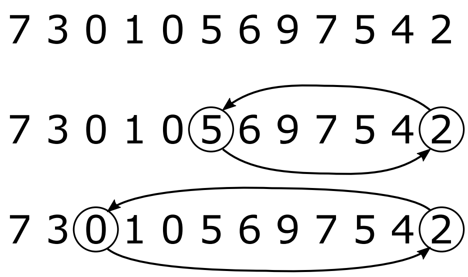
LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 10 páginas (não contando a folha de rosto), numeradas de 1 a 10. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 2 devem ser arquivos com sufixo *.py2*; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py3*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,input,print,write*
  - em Javascript: *scanf, printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Cinco

Nome do arquivo: "`cinco.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Considere um número decimal não divisível por 5. Queremos fazer exatamente uma operação de troca entre os dígitos de duas posições distintas para obter um número que seja divisível por 5. Quer dizer, precisamos escolher duas posições distintas e trocar os dígitos dessas duas posições. Mas queremos que o número resultante após a troca seja o maior número divisível por 5 possível.



Veja o exemplo da figura, 730105697542, que não é divisível por 5. Podemos fazer a primeira troca ilustrada e obter 730102697545, que é divisível por 5. Mas, se fizermos a segunda troca ilustrada na figura, vamos obter um número divisível por 5 ainda maior, 732105697540.

Dados os dígitos decimais de um número na entrada, não divisível por 5, seu programa deve imprimir os dígitos decimais do maior número divisível por 5 que pode ser obtido com exatamente uma troca de dígitos entre duas posições distintas. Caso não seja possível obter um número divisível por 5, imprima apenas -1.

Entrada

A primeira linha da entrada contém um inteiro N, indicando quantos dígitos decimais tem o número não divisível por 5. A segunda linha contém N inteiros D_i, $1 \leq i \leq N$, representando os dígitos decimais do número em questão.

Saída

Imprima uma linha contendo N inteiros representando os dígitos decimais do maior número divisível por 5 que pode ser obtido com exatamente uma troca de dígitos entre duas posições distintas. Caso não seja possível obter um número divisível por 5, imprima apenas -1.

Restrições

- $2 \leq N \leq 1000$
- D_i é um inteiro entre 0 e 9, inclusive.

Exemplo de entrada 1	**Exemplo de saída 1**
`12` `7 3 0 1 0 5 6 9 7 5 4 2`	`7 3 2 1 0 5 6 9 7 5 4 0`

Exemplo de entrada 2	**Exemplo de saída 2**
`5` `7 4 1 2 9`	`-1`

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

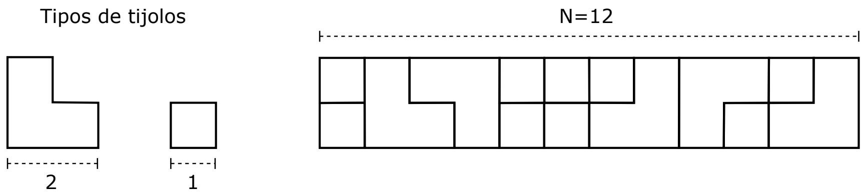
Exemplo de entrada 4	**Exemplo de saída 4**
`10` `6 5 0 5 0 4 5 3 4 4`	`6 5 4 5 0 4 5 3 4 0`

Exemplo de entrada 5	**Exemplo de saída 5**
`7` `9 7 4 5 3 5 2`	`9 7 4 5 3 2 5`

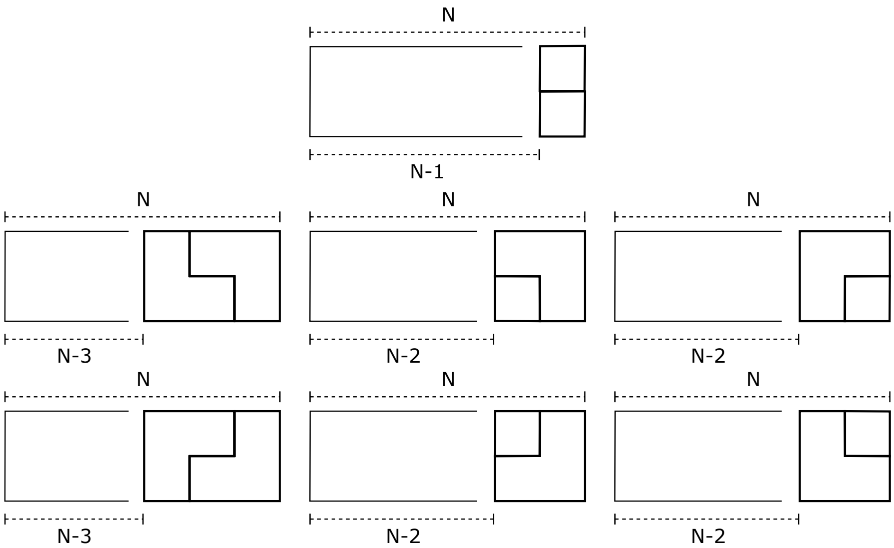
Muro

Nome do arquivo: "`muro.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Nós temos dois tipos de tijolos, como mostrado na parte esquerda da figura abaixo. A ideia é construir uma mureta de altura 2 e comprimento N. A parte direita da figura ilustra uma forma de usar os dois tipos de tijolos para construir uma mureta de comprimento $N = 12$.



Precisamos saber quantas formas distintas existem de construir a mureta com esses dois tipos de tijolos. Para isso, já temos duas observações: qualquer mureta de comprimento N vai terminar de uma das sete maneiras ilustradas abaixo e; o número de formas distintas de construir uma mureta de comprimento 2, 1 e 0 é, respectivamente, 5, 1 e 1 (Sim! Existe uma forma de construir a mureta de comprimento 0: usar nenhum tijolo).



Dado N, seu programa deve computar o número de formas distintas de construir uma mureta de comprimento N. Como esse número pode ser muito grande, seu programa deve imprimir o resto da divisão dele por $10^9 + 7$.

Entrada

A única linha da entrada contém um inteiro N, representando o comprimento da mureta.

Saída

Imprima uma linha contendo um inteiro, o número de formas distintas de construir a mureta com os dois tipos de tijolos. Imprima o resto da divisão desse número por $10^9 + 7$.

Restrições

- $0 \leq N \leq 10^4$.

Informações sobre a pontuação

- Para um conjunto de casos de teste valendo 20 pontos, $N \leq 16$.

Exemplo de entrada 1	**Exemplo de saída 1**
`2`	`5`

Exemplo de entrada 2	**Exemplo de saída 2**
`11`	`36543`

Exemplo de entrada 3	**Exemplo de saída 3**
`6`	`241`

Exemplo de entrada 4	**Exemplo de saída 4**
`0`	`1`

Exemplo de entrada 5	**Exemplo de saída 5**
`8712`	`844673301`

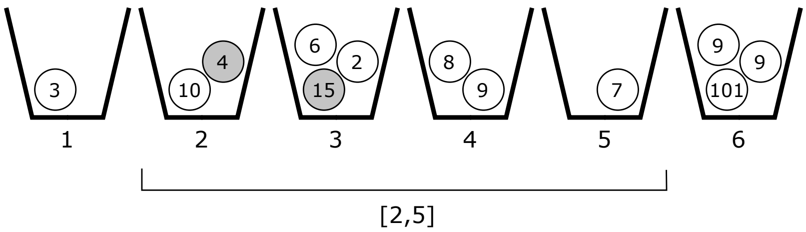
Baldes

Nome do arquivo: "`baldes.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Temos uma sequência de N baldes, identificados de 1 até N, cada balde contendo inicialmente uma bola de peso inteiro positivo. Queremos realizar uma sequência de M operações de dois tipos possíveis:

1. Adicionar uma bola de peso p ao balde i;
2. Dados a e b, com $1 \leq a < b \leq N$, imprimir a maior diferença absoluta possível entre o peso de duas bolas, de baldes distintos, dentro do intervalo de baldes $[a, b]$.

Por exemplo, na figura abaixo, para $N = 6$, o resultado da operação do tipo 2 para o intervalo $[2, 5]$ é 11, correspondente às bolas 4 e 15, dos baldes 2 e 3 respectivamente. Existe uma diferença absoluta maior para as bolas 15 e 2, mas elas estão no mesmo balde, portanto, essa diferença não conta.



Entrada

A primeira linha da entrada contém dois inteiros, N e M, respectivamente, o número de baldes e o número de operações. A segunda linha da entrada contém N inteiros indicando o peso da bola contida em cada balde inicialmente. As M linhas seguintes descrevem, cada uma, uma operação. Se a operação é do primeiro tipo, a linha contém o número 1 seguido de dois inteiros, P e I, indicando o peso da bola a ser adicionada e o identificador do balde. Se a operação é do segundo tipo, a linha contém o número 2 seguido de dois inteiros, A e B, representando o intervalo $[A, B]$ de baldes.

Saída

Para cada operação do segundo tipo, imprima uma linha contendo a maior diferença absoluta possível entre o peso de duas bolas, de baldes distintos, dentro do intervalo em questão.

Restrições

- $2 \leq N \leq 10^5$;
- $1 \leq M \leq 10^5$;
- $1 \leq A < B \leq N$;
- O peso das bolas está entre 1 e 10^6;
- A entrada contém pelo menos uma operação do segundo tipo.

Informações sobre a pontuação

- Para um conjunto de casos de teste valendo 10 pontos, $N \leq 100$ e $M \leq 100$;
- Para um conjunto de casos de teste valendo 40 pontos, $N \leq 10^4$ e $M \leq 10^4$.

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

`10 5`
`3 9 12 4 20 5 7 15 9 10`
`1 1 5`
`1 33 8`
`2 6 9`
`1 15 2`
`2 1 7`
	`28`
	`17`

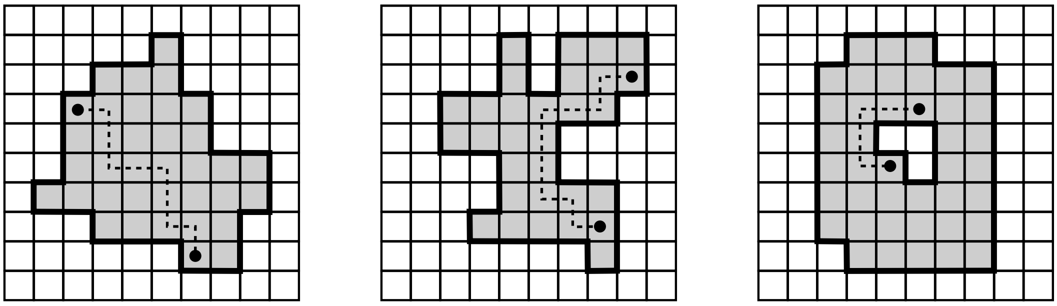
Exemplo de entrada 2	**Exemplo de saída 2**

`2 3`
`100 200`
`2 1 2`
`1 55 1`
`2 1 2`
	`100`
	`145`

Mancha

Nome do arquivo: "`mancha.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Juninho está participando de um projeto de iniciação científica sobre identificação de doenças de pele através de análises de imagens digitais. Muitas vezes o formato de uma lesão de pele, ou mancha, pode indicar as possibilidades de diagnóstico. O professor orientador tem algumas imagens digitalizadas de manchas e precisa identificar aquelas que são "regulares" segundo uma definição bastante precisa, que será dada abaixo. Juninho precisa da sua ajuda para processar a imagem da mancha e decidir se ela é ou não regular.



A imagem é um reticulado de $N \times N$ pixels. Os pixels escuros representam a mancha, que é sempre conexa, ou seja, é composta de apenas uma componente. De forma mais precisa, dado qualquer par de pixels pertencentes à mancha, sempre existe um caminho, uma sequência de pixels escuros entre eles seguindo somente por direções ortogonais, totalmente contido dentro da mancha. A figura acima ilustra três possíveis manchas, para $N = 10$.

Dados dois pixels P e Q, a distância de Manhattan entre eles é definida como: $d_{manhattan}(P,Q) = |P_l - Q_l| + |P_c - Q_c|$, onde P_l é o índice da linha do pixel P e P_c é o índice da coluna do pixel P, na imagem digitalizada. O mesmo vale para Q_l e Q_c. Ou seja, a distância de Manhattan é a soma da diferença absoluta entre a linha de P e a linha de Q com a diferença absoluta entre as colunas de P e Q. Dados dois pixels P e Q que pertencem à mancha, definiremos $d(P,Q)$ como sendo o comprimento do menor caminho existente entre P e Q, que esteja totalmente contido dentro da mancha.

No exemplo da figura mais à esquerda, onde P e Q estão representados por um pequeno círculo, $d(P,Q) = 9$ e $d_{manhattan}(P,Q) = 9$. Na figura do meio, $d(P,Q) = 10$ e $d_{manhattan}(P,Q) = 6$; e na figura mais à direita, $d(P,Q) = 5$ e $d_{manhattan}(P,Q) = 3$.

Finalmente, uma mancha será *regular* se, para qualquer par de pixels P e Q pertencentes à mancha, tivermos $d(P,Q) = d_{manhattan}(P,Q)$. Dessa forma, verifique que a figura mais à esquerda ilustra uma mancha regular, enquanto que as outras duas são irregulares.

Entrada

A primeira linha da entrada contém um inteiro N, representando as dimensões da imagem. As N linhas seguintes contêm, cada uma, uma cadeia de N caracteres definindo uma linha de pixels da imagem. Os caracteres podem ser: "`.`" para pixels fora da mancha; e "`*`" para pixels que pertencem à mancha.

Saída

Imprima uma linha contendo o caractere "`S`", se a mancha for regular; ou "`N`", se for irregular.

Restrições

- $2 \leq N \leq 1000$;
- A mancha possui pelo menos dois pixels.

Informações sobre a pontuação

- Para um conjunto de casos de teste valendo 20 pontos, $N \leq 20$;
- Para um conjunto de casos de teste valendo 40 pontos, $N \leq 100$.

Exemplo de entrada 1

```
10
..........
.....*....
...***....
..*****...
..*****...
..*******.
.********.
...*****..
......**..
..........
```

Exemplo de saída 1

```
S
```

Exemplo de entrada 2

```
10
..........
....*.***.
....*.***.
..******..
..****....
....**....
....****..
...*****..
.......*..
..........
```

Exemplo de saída 2

```
N
```

Exemplo de entrada 3

```
2
.*
**
```

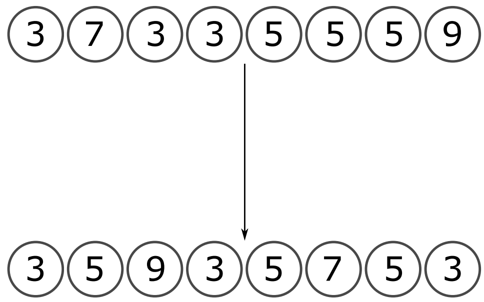
Exemplo de saída 3

```
S
```

Bolas

Nome do arquivo: "`bolas.x`", *onde* `x` *deve ser* `c|cpp|pas|java|js|py2|py3`

Temos oito bolas, colocadas lado a lado em uma sequência. Cada bola tem um número impresso, que pode ter valor de 0 até 9. Queremos trocar algumas bolas de posição na sequência de modo que nenhum par de bolas vizinhas na sequência tenha o mesmo número. Quer dizer, não pode haver duas bolas, uma ao lado da outra, com o mesmo número. A figura ao lado mostra um exemplo para o qual isso foi possível. Mas será que sempre é possível? Seu programa deve decidir se é ou não possível obter uma sequência em que não haja bolas vizinhas com o mesmo número.



Entrada

A única linha da entrada contém uma sequência de oito inteiros B_i, para $1 \leq i \leq 8$, representando os números impressos em cada bola da sequência.

Saída

Imprima uma linha contendo o caractere "`S`" se for possível trocar bolas de posição e obter a sequência sem bolas vizinhas com o mesmo número; ou o caracter "`N`" se não for possível.

Restrições

- B_i é um inteiro entre 0 e 9, inclusive.

Exemplo de entrada 1	Exemplo de saída 1
`3 7 3 3 5 5 5 9`	`S`

Exemplo de entrada 2	Exemplo de saída 2
`8 3 8 8 8 8 8 0`	`N`