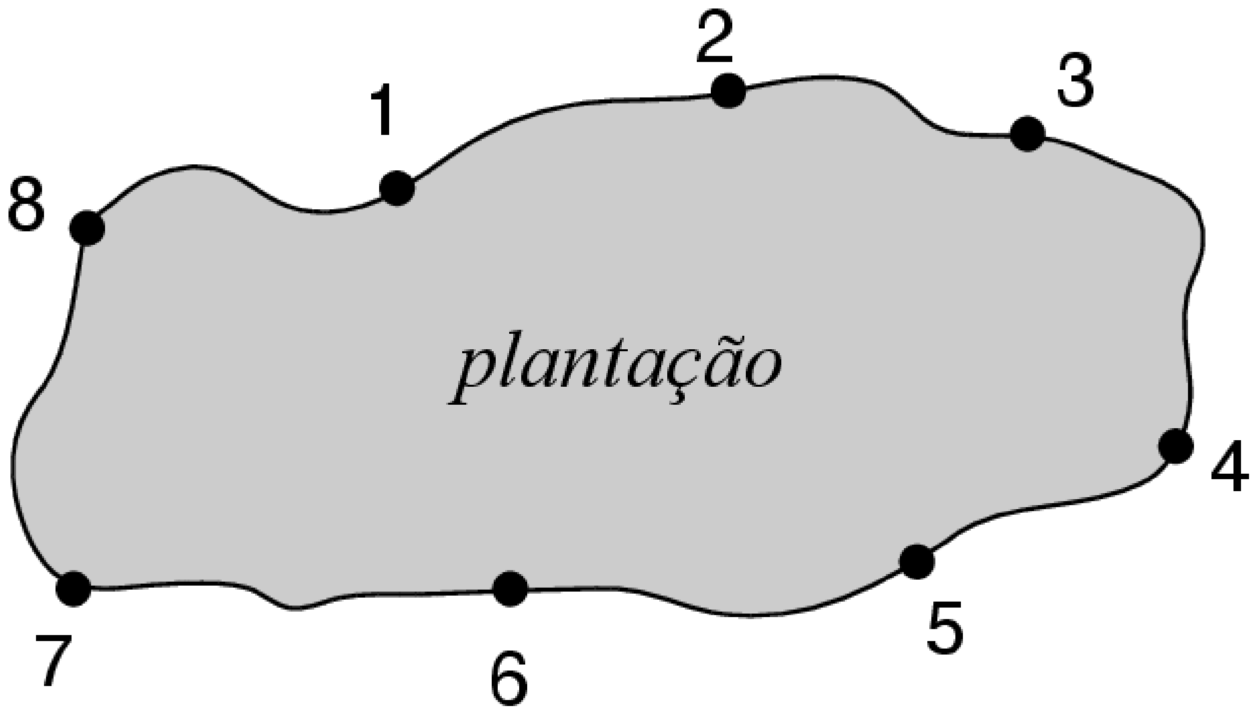
Algoritmos e Programação I

Lista de Exercícios da OBI

1. Robô

Um fazendeiro comprou um robô-espantalho para espantar os pássaros de sua plantação de milho. O robô se move ao longo de um caminho que circunda a plantação. No caminho há N estações numeradas sequencialmente, a partir de 1, no sentido horário. A figura abaixo mostra um exemplo com oito estações.



O robô inicia cada dia na estação número 1, e então obedece a uma sequência de comandos. Os comandos são gerados por um algoritmo de aprendizagem de máquina que coleta informações através de sensores espalhados na plantação, para garantir uma cobertura de vigia máxima. Cada comando faz com que o robô se mova para outra estação, vizinha à estação em que ele se encontra, ou no sentido horário ou no sentido anti-horário. O robô permanece nessa nova estação até receber um novo comando.

Apesar da promessa de que o robô protegeria a plantação, ao final de um determinado dia o fazendeiro notou que parte de sua plantação estava devastada por pássaros. O fazendeiro agora quer entender melhor o que aconteceu.

Dados o número da estação mais próxima à área devastada e a sequência de comandos que o robô obedeceu naquele dia, escreva um programa para determinar quantas vezes o robô permaneceu na estação mais próxima à área devastada.

Entrada

A primeira linha contém três inteiros N, C e S, representando respectivamente o número de estações, o número de comandos e o número da estação mais próxima à área devastada. A segunda linha contém C inteiros $X_1, X_2, \ldots, X_C$, representando a sequência de comandos recebidos pelo robô. Para i = 1, 2, …, C, se X_i é 1 então o i-ésimo comando significa "mova-se para a próxima estação no sentido horário", enquanto se X_i é -1 então o i-ésimo comando significa "mova-se para a próxima estação no sentido anti-horário". O robô sempre inicia na estação número 1.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de vezes que o robô permaneceu na estação número S durante o dia.

Restrições

- $2 \leq N \leq 100$
- $1 \leq C \leq 1000$

Exemplos

Entrada	Saída
8 8 3 1 -1 1 1 1 -1 1 1	2

Entrada	Saída
5 4 1 1 1 1 1	1

Entrada	Saída
2 1 1 1	1

Entrada	Saída
2 1 2 1	1

Entrada	Saída
2 2 1 -1 1	2

Entrada	Saída
2 2 1 -1 -1	2

2. Cálculo rápido

Algumas pessoas conseguem fazer cálculos matemáticos com uma velocidade impressionante. Laurinha tem essa habilidade! Um cálculo que ela consegue fazer muito rapidamente é: dados três números inteiros S, A, e B, determinar quantos números do intervalo [A, B] têm a soma de seus dígitos igual a S.

Por exemplo, se S = 3, A = 10 e B = 30, então a reposta é 3, pois existem três números no intervalo [10,30] cuja soma dos dígitos é igual a três: 12, 21 e 30.

Sua tarefa é escrever um programa de computador para, dados os três números, tentar calcular a resposta mais rapidamente do que Laurinha consegue.

Entrada

A primeira linha da entrada contém um número inteiro S, o valor da soma dos dígitos. A segunda e a terceira linhas contêm respectivamente os inteiros A e B.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, quantos números no intervalo dado têm a soma de dígitos indicada.

Restrições

- $1 \leq S \leq 36$

- $1 \leq A \leq 10\,000$
- $1 \leq B \leq 10\,000$
- $A \leq B$

Exemplos

Entrada	**Saída**
3 10 30	3

Entrada	**Saída**
15 1 20	0

Entrada	**Saída**
1 1 10000	5

3. Potência

A profa. Vilma preparou uma tarefa de programação sobre a *operação de potenciação*. Para lembrar, seja um número real n e um número inteiro p igual ou maior do que zero, então a operação de potenciação n^p tem o valor de n multiplicado por ele mesmo p vezes (se p = 0 o resultado da operação de potenciação é 1). Por exemplo, $2^3 = 2 \times 2 \times 2 = 8$ e $102^0 = 1$.

A tarefa preparada pela profa. Vilma foi a seguinte: *Escreva um programa para calcular o valor das seguintes expressões contendo operações de potenciação:*

$$2^4 + 12^3 + 300^3 + 15^2 + 4^2$$

Veja que cada termo das expressões tem a forma n^p onde n e p são números inteiros e p tem apenas um dígito decimal.

No entanto, quando a profa. Vilma colocou o enunciado da tarefa na Internet, a formatação do enunciado foi corrompida, fazendo com que as expressões aparecessem assim para os alunos:

$$24 + 123 + 3003 + 152 + 42$$

Note que por exemplo 2^4 virou 24, 12^3 virou 123, 300^3 virou 3003 e assim por diante, ou seja, as operações de potenciação desapareceram!

Nesta tarefa, você deve escrever um programa para calcular o valor das expressões da tarefa original da profa. Vilma, sabendo que a formatação do enunciado foi corrompida conforme explicado acima.

Entrada

A primeira linha da entrada contém um número inteiro N, o número de termos da expressão. Cada uma das N linhas seguintes contém um inteiro T_i, indicando um termo da expressão com formatação corrompida.

Saída

Seu programa deve produzir uma linha, contendo um único número inteiro, o valor da soma dos termos da expressão, sabendo que a formatação dos termos foi corrompida como explicado acima.

Restrições

- $1 \le N \le 10$
- $10 \le T_i \le 9999$ para $1 \le i \le N$
- O resultado é menor do que 10^9.

Exemplos

Entrada	**Saída**
2 24 123	1744

Entrada	**Saída**

3 303 152 42	27241

Entrada	Saída
6 12 13 24 20 31 43	86

4. Progressões Aritméticas

Bob é um aluno do ensino médio que gosta muito de matemática. Na última aula ele aprendeu o que são *Progressões Aritméticas* (*PAs*) e ficou fascinado por elas. Pelo que Bob entendeu, Progressões Aritméticas são sequências de números nas quais a diferença entre dois elementos consecutivos é sempre igual a uma constante *r*, chamada de razão da PA.

Um exemplo de Progressão Aritmética de razão 2 é -1, 1, 3, 5. Além disso, toda sequência com um ou dois elementos é sempre uma Progressão Aritmética. Por outro lado, 5, 6, 8, 9, 10 não é uma PA porque a diferença entre elementos consecutivos não é constante: a diferença entre os dois primeiros elementos é 6 - 5 = 1, enquanto a diferença entre o terceiro e o segundo elementos é 8 - 6 = 2.

Bob percebeu que qualquer sequência, mesmo que a mesma não seja uma Progressão Aritmética, pode ser quebrada em sequências menores que são PAs. Por exemplo, vimos que a sequência 5, 6, 8, 9, 10 não é uma PA, mas podemos quebrar ela entre o 6 e o 8 para obtermos as sequências 5, 6 e 8, 9, 10, que são PAs. Note que não existe como quebrar a sequência em menos partes se quisermos ter apenas PAs no fim do procedimento.

Bob é fascinado por programação mas ainda não sabe programar muito bem, e por isso pediu sua ajuda: ele não está conseguindo descobrir como quebrar sequências muito grandes de um jeito eficiente; por isso, pediu que você escrevesse um programa para, dada uma sequência qualquer, imprimir o número mínimo de partes em que precisamos quebrar a sequência para termos apenas Progressões Aritméticas no término do processo. Caso a sequência original já seja uma PA, podemos terminar o processo com uma única parte, e portanto a resposta para esse caso é 1.

Entrada

A primeira linha da entrada é composta por um inteiro N , o número de elementos da sequência. Na segunda linha existem N inteiros a_i, os elementos da sequência.

Saída

A saída deve conter uma única linha, indicando o número mínimo de partes em que Bob precisa quebrar a sequência original para que ele termine apenas com PAs.

Restrições

- $1 \leq N \leq 10^5$
- $-10^5 \leq a_i \leq 10^5$

Exemplos

Entrada	**Saída**
5 1 2 3 4 5	1
Entrada 7 -2 0 2 3 3 4 6	**Saída** 3

É fácil verificar que a sequência -2, 0, 2, 3, 3, 4, 6 (do exemplo acima) não é uma PA, pois 2 - 0 ≠ 3 - 2. Verificando manualmente, você pode constatar que não é poss&?vel particionar a sequência em duas de tal forma que ambas as partes sejam PAs. Entretanto, existe uma maneira de particionar a sequência em 3 PAs:

-2, 0, 2	3, 3	4, 6

Portanto, temos que a resposta para este exemplo é 3.

Entrada	**Saída**
4 -2 0 3 6	2

A sequência -2, 0, 3, 6 (do exemplo acima) pode ser particionada de várias formas. As únicas maneiras que resultam em PAs são as seguintes:

- Com 4 partes temos 1 possibilidade:

-2	0	3	6

- Com 3 partes temos 3 possibilidades:

-2,0	3	6
-2	0,3	6
-2	0	3,6

- Com 2 partes temos 2 possibilidades:

-2,0	3,6
-2	0,3,6

5. Maratona

A maratona é talvez a prova mais desgastante entre as modalidades olímpicas: são quarenta e dois mil, cento e noventa e cinco metros de percurso. Por isso, os organizadores sempre posicionam vários postos de água ao longo do trajeto da prova, onde copos de água são distribuídos aos competidores.

João Saci é um jovem atleta que tem boas chances de se tornar um maratonista de primeira linha. No entanto, João Saci descobriu que somente consegue terminar uma maratona se ingerir alguns copos de água durante o percurso. O Laboratório de Biomecânica da universidade local, através de experimentos, determinou que João Saci consegue percorrer exatamente mais dois mil metros após o instante em que ingere um copo de agua. A distância que João Saci consegue percorrer após ingerir um copo de água é denominada de *distância intermediária máxima*. Assim, se a distância entre dois postos de água consecutivos no percurso da maratona for sempre menor ou igual do que a distância intermediária máxima de João Saci, ele consegue terminar a prova. Caso contrário ele não consegue terminar a prova.

O Laboratório de Biomecânica quer agora realizar estudos similares com outros maratonistas, que têm valor de distâncias intermediárias máximas distintas, e precisa de sua ajuda.

Tarefa

Sua tarefa é escrever um programa que, dada a posição dos postos de água ao longo do percurso, e a distância intermediária máxima de um atleta, determine se o atleta consegue ou não completar a prova.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada contém dois números inteiros N e M , separados por um espaço em branco, indicando respectivamente o número de postos de água ($2 \le N \le 10000$) e a distância intermediária máxima de um atleta, em metros ($1 \le M \le 42195$). A segunda linha contém N números inteiros P_i , separados por um espaços em branco, representando a posição dos postos de água ao longo do trajeto da maratona. A posição de um posto de água é dada pela distância, em metros, do início do percurso até o posto de água ($0 \le P_i \le 42195$ para $1 \le i \le N$). O primeiro posto de água está sempre localizado no ponto de partida (ou seja, $P_1 = 0$) e todos os postos estão em posições distintas. Além disso, os postos de água são dados na ordem crescente de sua distância ao início do percurso.

Note que a distância total da prova é a oficial para a maratona, ou seja, 42195 metros.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha contendo o caractere "S" se o atleta consegue terminar a prova, ou o caractere "N" caso contrário.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \le 100$.
- Em um conjunto de casos de teste que totaliza 70 pontos, $N \le 2000$.

Exemplos

Entrada	Saída
3 20000 0 20000 33333	S
Entrada 8 6000 0 6000 12000 18000 24000 32000 37000 40000	**Saída** N