V Maratona Mineira de Programação

Caderno de Problemas — Competição Principal

Uberaba, 04 de Junho de 2016

Instruções:

- Este caderno contém 13 problemas: as páginas estão numeradas de 1 a 18, não contando esta folha de rosto. Verifique se o caderno está completo.
- Em todos os problemas, a entrada do programa deve ser lida na *entrada padrão*. A saída deve ser escrita na *saída padrão*.

Problema A. Ordenando para o Rodolfo

Arquivo-fonte: `ordenacao.c`, `ordenacao.cpp` ou `ordenacao.java`

Rodolfo é professor de algoritmos há muitos anos. Infelizmente, os códigos que ele usava para corrigir os trabalhos práticos não compilam em seu novo computador, e, por isso, ele pediu sua ajuda para escrever um novo corretor automático! Uma parte desse corretor deve determinar se uma sequência de 5 inteiros está ordenada de forma crescente. Rodolfo pediu para você fazer essa parte primeiro. Mãos à obra!

Entrada

A entrada consiste de uma única linha contendo 5 inteiros X_i, separados por um único espaço.

Saída

A saída consiste de uma única linha, contendo "SIM"(sem aspas) se a sequência estiver ordenada de forma crescente, ou "NAO", caso contrário.

$-100000 \leq X_i \leq 100000$

Exemplos

Entrada	Saída
1 2 3 4 5	SIM

Entrada	Saída
5 4 3 2 1	NAO

Problema B. Andando de Bicicleta

Arquivo-fonte: `bicicleta.c`, `bicicleta.cpp` ou `bicicleta.java`

Gabriel trabalha em uma grande empresa de tecnologia, dessas com o escritório alternativo. Ele usa uma bicicleta que ganhou da empresa para ir e voltar do trabalho. Começou a andar tanto de bicicleta que está pensando em participar de competições e maratonas de bicicleta. Para isso, ele quer saber a velocidade média que ele consegue atingir quando vai e volta do trabalho.

Um belo dia, Gabriel comprou um velocímetro que mostrava a velocidade média que ele fazia em um determinado percurso. Verificou uma velocidade média de V_a km/h no percurso de casa para o trabalho. Já na volta, verificou que a velocidade média foi de V_b km/h. Eis que ele ficou com uma dúvida: qual a velocidade média total obtida considerando todo o percurso (ida e volta)?

Entrada

Composto por uma única linha com dois números reais V_a e V_b com uma casa decimal cada, referentes as velocidades médias verificadas por Gabriel.

Saída

Um número real com duas casas decimais correspondente a velocidade média total do percurso inteiro.

$0.1 \leq V_a, V_b \leq 60.0$

Exemplos

Entrada	Saída
10.0 10.0	10.00

Entrada	Saída
5.0 5.0	5.00

Problema C. Não é a mamãe!

Arquivo-fonte: `mamae.c`, `mamae.cpp` ou `mamae.java`

Beibe Sauro tinha uma grande dificuldade em chamar o seu pai de ”Papai”, e o chamava de ”Não é a mamãe”. Só que o tempo passou, e seus filhos, netos, bisnetos e toda sua descendência tinham a mesma dificuldade. Então, sua tarefa é, dada uma estrutura de árvore genealógica, dizer se um dinossauro é ”Não é a mamãe”de um outro, ou seja, pai dele, ou ”não é o Não é a mamãe”caso contrário.

Entrada

A primeira linha da entrada contêm 2 inteiros, a quantidade R de relações de parentesco e a quantidade V de verificações, $1 \leq V, R \leq 1000$. Cada uma das próximas R linhas possui dois nomes, de modo que o primeiro é o nome do pai e o segundo é o nome do filho na relação determinada. Por fim, as próximas V linhas tem dois nomes, o qual seu programa deve verificar se o primeiro é pai do segundo de acordo com as R relações.

Cada nome tem no máximo 20 caracteres do alfabeto contendo apenas letras.

Saída

Para cada verificação, deve imprimir uma linha informando o resultado da verificação, ”Nao e a mamae”, caso exista a relação respectiva, ou ”nao e o Nao e a mamae”, caso não exista.

Exemplos

Entrada	Saída
6 5 dino beibe dino bob beibe neto beibe teno neto bisneto bob bobinho dino beibe beibe dino beibe neto neto bisneto bobinho bob	Nao e a mamae nao e o Nao e a mamae Nao e a mamae Nao e a mamae nao e o Nao e a mamae

Problema D. Indicador Multifatorial de Ameaça Dinossáurica

Arquivo-fonte: `ameaca.c`, `ameaca.cpp` ou `ameaca.java`

O paleontólogo Wenry Halton "Indione"Janas vem estudando dinossauros desde que se entende por gente, e chegou a uma fórmula para calcular o Indicador Multifatorial de Ameaça Dinossáurica (IMAD):

$$IMAD(S,A,C,P) = \left|\sum_{k=1}^{T(S)} P^{1/k}\cos k\pi\right| \cdot \binom{max(V(S),C(S))+1}{min(V(S),C(S))-1} \cdot \left(\frac{\left\lceil\sqrt{V(S)^e+C(S)^e}\right\rceil}{\lfloor\pi+\ln(1+A\cdot C)\rfloor}\right)$$

P = peso do dinossauro (em quilogramas)
S = nome do dinossauro
$T(S)$ = quantidade de letras no nome do dinossauro
$V(S)$ = quantidade de vogais no nome do dinossauro ('A', 'E', 'I', 'O' e 'U' são as vogais consideradas, podendo ser maiúsculas ou minúsculas)
$C(S)$ = quantidade de consoantes no nome do dinossauro ('K', 'W' e 'Y' também são considerados consoantes)
A = altura do dinossauro (em metros)
C = comprimento do dinossauro (em metros)
π = 3.14159265358979, e = 2.71828182845905
$\lfloor x\rfloor$ e $\lceil x\rceil$ significam, respectivamente, o valor de x arredondado para baixo (ou seja, a parte inteira de x) e para cima (ou seja, o menor inteiro maior ou igual a x).

Em posse dos dados sobre diversos dinossauros, sua tarefa é analisa-los e ordena-los, baseando-se no Indicador Multifatorial de Ameaça Dinossáurica.

Entrada

Inicia com um inteiro N ($1 \leq N \leq 51$), que indica a quantidade de dinossauros a serem informados. A seguir, há N linhas contendo o nome do dinossauro S (sem espaços e com, no máximo, 20 caracteres, apenas letras maiúsculas e minúsculas sem acento) e três números reais A, C ($1 \leq A \leq B \leq 10^2$) e P ($1 \leq P \leq 10^5$), representando, respectivamente, a altura, o comprimento e o peso do dinossauro em questão.

Saída

Imprima o nome dos dinossauros, ordenados de forma crescente de acordo com seu IMAD. Também, na frente de cada nome, inclua uma quantidade de exclamações igual ao número de dígitos presentes na parte inteira do IMAD deste dinossauro. Em caso de empate entre IMADs, ordene os nomes de forma alfabética crescente.

Exemplos

Entrada	Saída
2 Ornithomimus 2.4 4.6 136 Gallimimus 2.4 5.5 118	Gallimimus!!!!! Ornithomimus!!!!!!

Entrada	Saída
3 Kentrosaurus 1.5 5.2 1814 Coelophysis 0.9 2.7 46 Diplodocus 7.3 27.1 22680	Coelophysis!!!!! Kentrosaurus!!!!!!! Diplodocus!!!!!!!

Problema E. Árvore Geneopaleontológica

Arquivo-fonte: `argen.c`, `argen.cpp` ou `argen.java`

A região de Sinodaurolândia, um entreposto entre Galpones e Yendegaia (no sul do Chile), tem atraído cada vez mais a atenção de paleontólogos. Isto porque a quantidade de fósseis que vem sendo descobertos na região superam os números de qualquer outro lugar na Terra, configurando-o como um sítio arqueológico promissor e de valor incalculável.

Entretanto, devido à enorme quantidade de dados coletados pelos pesquisadores, um clima de euforia tomou conta da região, entorpecendo todos os indivíduos com ilusões de grandeza e com novas teorias evolucionistas nada ortodoxas. Mas o pior problema é o comprometimento da capacidade de catalogação de informação no dia-a-dia de trabalho.

Para tentar botar ordem na casa, ou seja, organizar toda a informação obtida até o momento, o Comitê Central de Catalogação Paleontológica Operacional (C3PO) enviou um representante cuja única atribuição será a organização do material, já que o esquecido intuito da pesquisa era estabelecer relações entre diferentes espécies encontradas na região.

Entrada

A entrada inicia com um inteiro N ($1 \leq N \leq 10^2$), que indica o número de anotações que deverão ser analisadas. Cada anotação é composta por duas *strings* separadas por espaço (sem espaços e com, no máximo, 50 caracteres), contendo o nome de um dinossauro e o nome de seu ancestral imediato.

Saída

Imprima os nomes de todos os dinossauros que compõem a árvore genealógica formada, desde o ancestral mais remoto até o mais recente.

Observações

Cada espécie na entrada possui máximo de 1 ascendente. Apenas uma espécie na entrada não possui descdentes.

Exemplos

Entrada	Saída
2 Microvenator Saurornithoides Saurornithoides Nodosaurus	Nodosaurus Saurornithoides Microvenator

Entrada	Saída
3 Diplodocus Euoplocephalus Pachycephalosaurus Supersaurus Supersaurus Diplodocus	Euoplocephalus Diplodocus Supersaurus Pachycephalosaurus

Entrada	Saída
4 Kentrosaurus Argentinosaurus Saurornithoides Styracosaurus Styracosaurus Kentrosaurus Velociraptor Saurornithoides	Argentinosaurus Kentrosaurus Styracosaurus Saurornithoides Velociraptor

Problema F. Fuga da Mansão

Arquivo-fonte: `fuga.c`, `fuga.cpp` ou `fuga.java`

Ivan prefere jogar video-game ao invés de fazer os trabalhos da faculdade ou estudar para a maratona de programação. Um de seus jogos favoritos é "Rodent Evil 1". Nesse jogo você pode controlar um dentre 2 personagens disponíveis, Cristiane (abreviado por Cris) ou Gilson (abreviado por Gil) - o objetivo do jogo é escapar de uma mansão cheia de roedores zumbis.

Para dificultar a missão do jogador, existem diversos obstáculos e portas trancadas na mansão. Para destrancar uma porta é necessário ter a chave dessa porta, mas depois de destrancada não é mais necessária a chave para abrir essa porta. Para piorar, a personagem não consegue carregar muitos itens de uma vez, assim, ela pode deixar os itens em baús espalhados pela mansão para liberar espaço para pegar outros itens. Um item deixado em um baú é acessível, magicamente, em qualquer outro baú da mansão.

As chaves das portas estão espalhadas pela mansão; ao achar uma chave é possível pegá-la ou não, mas caso possua uma chave no inventário só é possível deixá-la em algum baú ou continuar segurando ela.

Ivan joga "Rodent Evil 1"há muito tempo, tanto tempo que ele memorizou o mapa da mansão, um grid NxM, incluindo a posição dos obstáculos, baús, chaves e portas. Por ser um jogo antigo, os personagens só conseguem se mover para posições adjacentes, formalmente, se a personagem está na posição (i, j) da mansão, ele consegue ir para as posições $(i + 1, j)$, $(i - 1, j)$, $(i, j + 1)$, $(i, j - 1)$ desde que essas posições estejam dentro da mansão e não possuam obstáculos. Uma porta trancada funciona como um obstáculo caso não se possua a chave dela no inventário e uma porta destrancada (ou caso se possua a chave dela) funciona como uma posição livre. É necessário estar na mesma posição da porta para poder destrancá-la. Também é necessário estar na mesma posição de uma chave para pegá-la e estar na mesma posição de um baú para poder usá-lo (segundo o mapa). Ivan é interessado em speedruns - isto é, terminar o jogo no menor tempo possível. Você pode indicar qual o tempo mínimo para que isso seja feito ou se é impossível terminar? Colocar ou retirar algum item de algum baú, pegar alguma chave ou abrir alguma porta não gastam tempo mas cada vez que o personagem se movimenta para alguma posição adjacente é gasta uma unidade de tempo. Ivan é muito bom nesse jogo, então ele não se preocupa se há roedores zumbis em seu caminho.

Entrada

A entrada começa com uma linha com 3 inteiros, N, M, T, $1 \leq N, M \leq 9$, $1 \leq T \leq 4$ as dimensões da mansão e a quantidade de itens que o personagem consegue carregar. Cada uma das N linhas seguintes contém M caracteres cada, a representação da mansão no mapa. No mapa o caractere '.' indica posições livres. O caractere '#' indica posições com obstáculos. Os caracteres minúsculos 'c', 'd', 'h', 's' indicam chaves. Há no máximo uma chave de cada tipo por mapa. Os caracteres maiúsculos 'C', 'D', 'H', 'S' indicam portas trancadas. Uma porta de caractere 'C' só pode ser destrancada por uma chave 'c', uma porta 'D' por uma chave 'd' e assim por diante. O caractere 'I' indica a posição inicial de Cris ou Gil. O caractere 'E' indica a saída da mansão. O caractere 'B' indica um baú.

Para cada entrada é garantido que $|I| = 1$, $|E| = 1$, $|B| \leq 4$, $|C| + |D| + |H| + |S| \leq 8$, onde $|x|$ indica quantas vezes o caractere 'x' aparece na entrada.

Saída

Para cada entrada imprima uma linha com o tempo mínimo para fugir da mansão. Caso seja impossível fugir da mansão imprima -1

Exemplos

Entrada	Saída
1 7 3 IsSSSSE	6

Entrada	Saída
4 3 1 IdD ##. #B. ECc	9

Pegamos a chave 'd', abrimos a porta 'D',colocamos essa chave no baú 'B', pegamos a chave 'c', abrimos a porta

'C' e chegamos à saída.

Entrada	Saída
4 3 2 IdD ##. #B. ECc	7

Nesse caso não precisamos usar o baú já que conseguimos carregar 2 chaves.

Entrada	Saída
5 5 1 IsS## ##B## ##hHB ####S ####E	8

Pegamos a chave 's', abrimos a porta 'S' da primeira linha do mapa, guardamos a chave no baú da segunda linha do mapa, pegamos a chave 'h', abrimos a porta 'H', colocamos a chave 'h' no baú da terceira linha do mapa, pegamos a chave 's' desse mesmo baú, abrimos a porta 'S' trancada e chegamos à saída.

Entrada	Saída
1 7 1 I.dDhHE	-1

Só conseguimos carregar uma chave, então não conseguiremos abrir as 2 portas já que não há baús.

Problema G. DNA Dinossáurico

Arquivo-fonte: `dna.c`, `dna.cpp` ou `dna.java`

Um grupo de pesquisadores recentemente descobriu um método eficiente para sequenciar DNA Dinossáurico a partir de fósseis. Isto nos permite agora entender melhor esses animais tão fascinantes que habitaram nosso planeta antes de nós. Semelhante ao caso humano, o DNA Dinossáurico também era armazenado no núcleo da célula dinossáurica, levado ao citoplasma por RNA mensageiro dinossáurico e interpretado por ribossomos, que contém a codificação do código genético dinossáurico (que pode ser representado por uma tabela que mapeia uma sequência de aminoácidos de entrada para outra sequência de aminoácidos de saída) e produzem com ele enzimas cuja estrutura terciária é bastante interessante. Para sua sorte, nenhuma dessas informações é relevante para este problema.

Uma sequência de DNA Dinossáurico é basicamente uma sequência com 4 aminoácidos possíveis: cretacina (representado pela letra **C**), dinossaurina (representado pela letra **D**), fossilina (**F**), e tirrexina (**T**). Por exemplo, uma sequência possível de DNA Dinossáurico é **CDDFTFC**. **ACAATG** é DNA humano, certamente não dinossáurico.

O que é importante é que os pesquisadores descobriram uma propriedade do DNA do dinossauro muito correlacionada com quão perigoso ele era. Ou seja, dado o DNA de dois dinossauros e tempo suficiente, os pesquisadores sabem dizer qual deles sairia vencedor em uma partida de MMA Dinossáurico (ou, mais realisticamente, qual deles mais provavelmente teria o outro como presa). Isto é de extrema valia para entender os fatores que afetaram a sobrevivência de cada espécie.

Esta propriedade de ouro é a *palindromicidade* do DNA. Você já deve saber que uma sequência de caracteres é *palíndrome* se ela é a mesma se lida da esquerda para a direita ou da direita para a esquerda. Por exemplo, **DFTFD** é uma sequência palíndrome, mas **DFTTT** não é. Nem toda sequência de DNA é palíndrome; contudo, todas possuem um número associado que é sua *palindromicidade*. A palindromicidade de uma sequência de DNA s é o maior inteiro d que satisfaz as seguintes propriedades:

- d é um divisor de $|s|$;
- Se dividimos s em d subsequências contíguas de caracteres de igual tamanho $\frac{|s|}{d}$, a primeira subsequência é igual à última; a segunda é igual à penúltima, e assim por diante. Mais precisamente, se as subsequências são numeradas da esquerda para a direita de 1 a $\frac{|s|}{d}$, a subsequência de índice i é igual à subsequência de índice $\frac{|s|}{d} + 1 - i$.

Note que, por definição, toda cadeia de caracteres tem palindromicidade no mínimo 1, porque ela sempre pode ser dividida apenas em um bloco. Já a cadeia **DFDF** não é palíndrome, mas tem palindromicidade 2, porque se a divididimos em dois blocos de igual tamanho, eles são iguais (**DF** e **DF**). Uma cadeia de tamanho n que seja palíndrome (no sentido tradicional) tem palindromicidade n - podemos colocar cada caractere em seu próprio bloco.

Dada uma sequência de DNA dinossáurico, calcule a sua palindromicidade.

Entrada

A entrada consiste de uma única linha contendo uma sequência S de DNA dinossáurico, composta apenas pelos caracteres **C**, **D**, **F** e **T**.

Saída

Imprima uma única linha contendo um inteiro: a palindromicidade da sequência de DNA dinossáurico dada na entrada.

$$1 \leq |S| \leq 5 \times 10^5$$
$$S_i \in \{C, D, F, T\}$$

Exemplos

Entrada	Saída
CDCD	2

Entrada	Saída
CDFTFDC	7

Entrada	Saída
CDFTFDCC	1

Problema H. Happy Hour

Arquivo-fonte: `happyhour.c`, `happyhour.cpp` ou `happyhour.java`

Todo começo de semestre, o professor Pardal gosta de reunir os horários livres de cada aluno de sua turma de Estrutura de Dados e organizá-los para descobrir quais os melhores horários para fazer happy hour com a galera.

Apesar de ser um experiente professor de Computação, todo semestre ele combina esses horários de maneira manual. E se você construísse um programa para automatizar essa tarefa para o professor Pardal? Será que você conseguiria uns pontos extras?

Entrada

A entrada possui um inteiro A ($1 \leq A \leq 100$) que indica a quantidade de alunos na turma. Para cada uma das A linhas seguintes, existem 6 strings, separadas pelo caracter de espaço, representando os horários livres de um aluno em cada dia da semana (seg-sab). Uma string é formada por caracteres de A-L (sempre maiúsculos), onde cada letra indica um horário livre no respectivo dia da semana. Por dia, existem 12 horários possíveis (de A-L). Uma string sempre estará com seus caracteres em ordem lexicográfica.

Saída

A saída deve conter 6 strings (com seus respectivos caracteres em ordem lexicográfica), separadas por espaço, cada uma indicando os horários que toda a turma estará livre. Se não existe nenhum horário livre para um determinado dia, a string é vazia.

Exemplos

Entrada	Saída
3 ABEL ABIJKL AJKL ABEL JKL EFGHIJKL ABEFGL CIJKL ABH AKL ABIJKL EFGHIJKL ABL HIJKL AIL AFGJKL AFGJKL EFGHIJK	ABL IJKL A AL JKL EFGHIJK

Entrada	Saída
1 ABE KL ABEJKL ABEJKL AHL ABCDEFGHIJKL	ABE KL ABEJKL ABEJKL AHL ABCDEFGHIJKL

Problema I. Garçom da Maratona

Arquivo-fonte: `garcom.c`, `garcom.cpp` ou `garcom.java`

Como todos sabem, o mais importante na Maratona de Programação não é competir, mas sim comer! Por isso, vários pratos são servidos durante a competição. Para evitar que competidores precisem parar de programar para comer, garçons levam os pratos até as mesas dos competidores.

Assim que a competição acaba, todos os competidores deixam o local da competição em direção ao auditório onde acontecerá a cerimônia de encerramento. Loucos para descobrir qual foi o placar final, os competidores deixam os pratos que usaram em cima de suas mesas.

Existe um fileira com N mesas no local da competição. A i-ésima mesa nessa fileira contém uma pilha com A_i pratos. Os garçons estão se recusando a recolher os pratos pois estão todos espalhados. Eles só irão recolher K pilhas de pratos. Felizmente, é possível mover pratos. Em uma unidade de tempo, é possível mover um prato no topo da pilha de uma mesa para o topo da pilha de uma mesa adjacente. Isto é, em uma unidade de tempo, é possível remover um prato do topo da pilha da i-ésima mesa e colocá-lo no topo da pilha da $(i+1)$-ésima mesa ou da $(i-1)$-ésima mesa.

Movendo zero ou mais pratos de lugar, é possível colocar todos os pratos em, no máximo, K pilhas, de forma que os garçons possam recolhê-los. Determine o tempo mínimo necessário para colocar todos os pratos em, no máximo, K pilhas.

Entrada

A primeira linha da entrada possui dois inteiros: N, o número de mesas, e K, o número máximo de pilhas desejado. Em seguida, há uma linha contendo N inteiros, representando a quantidade de pratos em cada mesa. O i-ésimo desses inteiros, A_i, indica que a i-ésima mesa possui uma pilha com A_i pratos.

Saída

A saída deve conter uma única linha com o um inteiro indicando o tempo mínimo necessário para colocar todos os pratos em, no máximo, K pilhas.

$1 \leq N \leq 400$

$1 \leq K \leq N$

$1 \leq A_i \leq 10^5$

Exemplos

Entrada	Saída
4 2 1 1 1 1	2

Entrada	Saída
5 1 1 2 3 2 1	8

Problema J. Amigo Secreto

Arquivo-fonte: `amigo.c`, `amigo.cpp` ou `amigo.java`

Maria trabalha em uma fábrica de chocolates. Todos os anos, Maria e seus colegas de trabalho participam de um amigo secreto na festa de fim de ano da fábrica. Nesse amigo secreto, um sorteio é realizado para determinar o amigo secreto de cada participante, de forma que um participante nunca seja seu próprio amigo secreto. Na festa de fim de ano, um participante é sorteado para revelar seu amigo secreto e entregar um presente para ele. Toda vez que um participante que ainda não revelou seu amigo secreto recebe um presente, ele revela seu amigo secreto e entrega um presente para o mesmo. Se o último participante a receber um presente já revelou seu amigo secreto e ainda existem participantes que não revelaram, assim como no início, um participante que ainda não revelou seu amigo secreto é escolhido ao acaso para fazê-lo.

A memória de Maria é muito boa. Por isso, ela é desafiada por seus colegas de trabalho a se lembrar de alguma coisa frequentemente. No último desafio, os colegas de Maria pediram para ela listar os participantes do amigo secreto do fim de 2015 na ordem em que eles revelaram seus amigos secretos. Apesar de sua boa memória, Maria não se lembra da lista completa, mas ela se lembra de algumas posições da lista. Além disso, ela também se lembra do amigo secreto de cada participante.

Competitiva como é, Maria não quer perder o desafio. Ela acha que talvez seja possível recuperar a ordem em que os participantes revelaram seus amigos secretos, mas ela anda muito ocupada na fábrica de chocolates e não tem tempo para tentar resolver esse problema. Por isso, ela pediu a sua ajuda. Dados as posições da ordem de revelação que Maria se lembra e o amigo secreto de cada participante, se possível, determine a ordem em que os participantes revelaram seus amigos secretos.

Entrada

A primeira linha da entrada possui um inteiro N, o número de participantes do amigo secreto. Em seguida, há uma linha contendo N inteiros, representando o amigo secreto de cada participante. O i-ésimo desses inteiros, A_i, identifica o amigo secreto do participante identificado por i. Em seguida, há uma linha contendo N inteiros, representando as partes da ordem de revelação dos amigos secretos das quais Maria se lembra. O i-ésimo desses inteiros, R_i, identifica o participante que foi i-ésimo participante a revelar seu amigo secreto. Se $R_i = 0$, Maria não se lembra quem foi i-ésimo participante a revelar seu amigo secreto.

Saída

Caso seja possível determinar unicamente a ordem em que os participantes revelaram seus amigos secretos, a saída deve conter uma única linha com N inteiros separados por espaços. O i-ésimo desses inteiros deve identificar o i-ésimo participante a revelar seu amigo secreto. Caso não seja possível, a saída deve ser formada por uma única linha contendo o inteiro -1.

$1 \leq N \leq 100$

$1 \leq A_i \leq N$

$0 \leq R_i \leq N$

Exemplos

Entrada	Saída
5 5 1 4 3 2 3 0 0 0 5	3 4 2 1 5

Entrada	Saída
5 5 1 4 3 2 0 0 1 0 2	-1

Problema K. Hora da caça

Arquivo-fonte: `hora.c`, `hora.cpp` ou `hora.java`

Algumas espécies de dinossauros eram conhecidas por serem extremamente individualistas. Mas nem todas. Uma delas, os Nlogossauros, eram um exemplo de pensamento comunitário já no século 300000 A.C.

Os Nlogossauros se alimentavam de Quadradossauros. Os quadradossauros possuiam uma proteção tão forte que quando estavam protegidos era inviável cassá-los, mesmo para espécies com dentes avantajados, como era o caso dos Nlogossauros. Eles eram muito preguiçosos e medrosos e ficavam o dia todo parados ao lado de comida, e só se desprotegiam para se alimentar uma vez por dia, em um certo intervalo de tempo. Alguns pesquisadores suspeitam que o término da comida era o motivo de morte de muitos Quadradossauros, porque a preguiça de procurar outra fonte de comida era maior que sua vontade de viver.

Um grupo de Nlogossauros recentemente encontrou uma manada de Quadradossauros e está planejando um ataque no dia seguinte. Eles já observaram os Quadradossauros e sabem, para cada um, em que ponto no plano eles estão parados e em qual intervalo eles ficarão desprotegidos.

Para matar um Quadradossauro, um Nlogossauro precisa atacá-lo durante todo o intervalo de tempo em que ele está desprotegido. Ele sempre começa imobilizando o Quadradossauro subindo sobre ele (usando uma técnica desenvolvida em Gondwana), que então fica parado enquanto recebe ataques. Se o ataque não durar exatamente o mesmo período que o Quadradossauro fica desprotegido, ele será inútil, porque ao final do intervalo, se não estiver morto, o Quadradossauro consegue se proteger novamente, e de seu estado protegido nunca mais sairá por medo (ou seja, morre de fome porém protegido e neste estado é inútil para os Nlogossauros).

Para os fins deste problema, considere que a posição de cada dinossauro é um ponto no plano cartesiano. Para que o Nlogossauro mate o Quadradossauro, os dois precisam estar exatamente no mesmo ponto durante todo o intervalo em que o Quadradossauro fica desprotegido.

Os Nlogossauros vão se posicionar bem cedo para seu ataque, antes que qualquer um dos Quadradossauros esteja acordado. Assim, eles podem começar aonde quiserem. Tendo estudado dinossauros há bastante tempo, você conhece a velocidade com que cada dinossauro se move, e mover-se é a única tarefa que toma seu tempo. Sabendo o tamanho do bando de Nlogossauros e a posição de todos os Quadradossauros, bem como o intervalo em que estarão desprotegidos, sua tarefa é determinar se é possível que os Nlogossauros devorem todos os Quadradossauros em seu ataque.

Entrada

A primeira linha da entrada possui dois inteiros separados por um espaço: N, o número de Nlogossauros, e Q, o número de Quadradossauros. Em seguida, há Q linhas. A i-ésima dessas linhas contém quatro inteiros: x_i, y_i, b_i, e_i, onde (x_i, y_i) é a posição em que fica o i-ésimo Quadradossauro, enquanto que este dinossauro ficará desprotegido no intervalo de tempo $[b_i, e_i]$. As distâncias são dadas em metros e os limites do intervalo em segundos desde o início do dia. Todo Nlogossauros se move a uma velocidade constante de 1 metro por segundo.

Saída

Sua saída deve conter uma linha com um caractere: 'S' se um bando de N Nlogossauros, devidamente posicionados no início do dia e com alguma estratégia de ações de movimento e ataque planejados, pode devorar todos os Q Quadradossauros descritos, ou 'N' se isto é impossível.

$$1 \leq N, Q \leq 3000$$
$$0 \leq x_i, y_i, b_i, e_i \leq 10^9$$
$$e_i \geq b_i + 1$$

Exemplos

Entrada	Saída
1 2 0 0 0 1 1 0 2 3	S

Há um único Nlogossauro e dois Quadradossauros: um na posição (0, 0), vulnerável entre os tempos 0 e 1, e outro em (1, 0), vulnerável entre os instantes 2 e 3. O Nlogossauro pode tranquilamente começar em (0, 0), devorar

o primeiro Quadradossauro até o tempo $t = 1$, depois caminhar até o segundo Quadradossauro em 1 segundo, chegando lá em $t = 2$, e devorá-lo também até $t = 3$.

Entrada	Saída
1 3 0 0 0 1 1 0 1 2 0 1 2 3	N

Neste caso, além dos dois Quadradossauros do caso anterior, há um outro em $(1, 0)$ vulnerável durante o período $[1, 2]$. O único Nlogossauro não consegue, sozinho, devorar os três Quadradossauros.

Entrada	Saída
2 3 0 0 0 1 1 0 1 2 0 1 2 3	S

Agora há os mesmos três Quadradossauros do caso anterior, com um Nlogossauro a mais. Os dois podem devorar todo o bando de Quadradossauros. Basta que um devore o primeiro e segundo Quadradossauros da entrada, enquanto que o outro foque em devorar apenas o terceiro.

Problema L. Mouse Language

Arquivo-fonte: `mouse.c`, `mouse.cpp` ou `mouse.java`

Sabemos que o ensino de programação para crianças tem sido muito difundido nas escolas. Por isso, a Associação de Colégios Municipais de Minas Gerais, ACM-MG, criou uma linguagem de programação didática, a *Mouse Language*. Basicamente, nesta linguagem o programador controla um pequeno rato inicialmente colocado na origem de um plano cartesiano. Com um único comando, o programador é capaz de deslocar o rato para uma outra posição (ponto) no plano cartesiano. E sim, você leu corretamente, essa linguagem possui apenas **um comando**, definido por:

rotacione r ande p repita n

Onde r é o número em graus que o rato irá rotacionar em relação ao seu próprio eixo no sentido horário, p é a distância que será percorrida em linha reta pelo rato (translação) e n é o número de vezes que o rato deverá repetir a rotação e translação requeridas. É importante ressaltar que o rato desloca na direção que sua cabeça aponta, mudando de acordo com a rotação. Inicialmente, o rato aponta para a direção norte do eixo y, o vetor (0, 1). Perceba que em um só comando a Mouse Language já introduz o conceito de laços de repetição! Não é incrível?

O problema é que a ACM está muito ocupada e ainda não construiram um interpretador para a linguagem. Sabendo que iria acontecer a V Maratona Mineira de Programação, decidiram escolher o seu time para fazer o trabalho. Portanto, dado um programa escrito na Mouse Language, o seu interpretador deverá escrever a posição final do rato no plano cartesiano.

Entrada

A entrada é um programa escrito na Mouse Language. Cada linha possui exatamente um comando na linguagem, na forma descrita no problema. O valor r é um número em graus com uma casa decimal, enquanto p e n são números inteiros. A entrada termina com o final do arquivo.

Saída

Escreva a posição final do rato, um ponto no plano cartesiano, de coordenadas reais x e y com duas casas decimais cada.

$0.0 \leq r \leq 360.0$

$0 \leq p \leq 50$

$1 \leq n \leq 10^5$

O número de comandos em Mouse Language não ultrapassa 10^5.

Exemplos

Entrada	Saída
`rotacione 0.0 ande 3 repita 1` `rotacione 0.0 ande 1 repita 3`	`0.00 6.00`

Entrada	Saída

`rotacione 0.0 ande 5 repita 1`
`rotacione 90.0 ande 5 repita 1`
`rotacione 90.0 ande 5 repita 1`
`rotacione 90.0 ande 5 repita 1`
	`0.00 0.00`

Entrada	Saída
`rotacione 0.0 ande 5 repita 1` `rotacione 90.0 ande 5 repita 3`	`0.00 0.00`

Entrada	Saída
`rotacione 45.0 ande 8 repita 1` `rotacione 90.0 ande 8 repita 1`	`11.31 0.00`

Como podemos ver na figura 1, no primeiro caso teste. Inicialmente, o rato se encontra na origem, em direção norte ao eixo y, representado como o rato de número 0. Após a execução do primeiro comando, o programa o desloca como mostrado pela seta vermelha, chegando na posição representada pelo rato de número 1. O segundo comando o desloca em um ponto, porém três vezes, como indicado pelas demais setas vermelhas na figura. O rato numerado por 2, portanto, é a posição final que será a saída do programa.

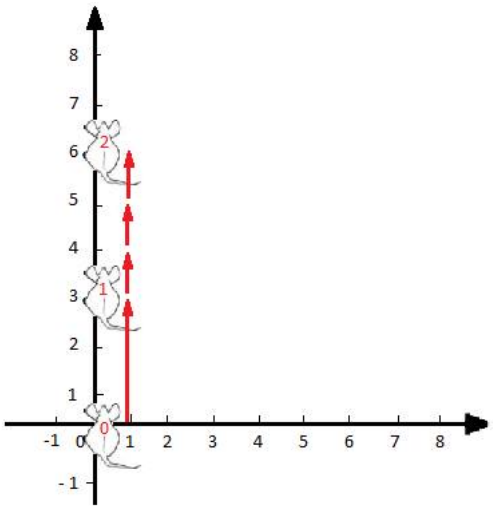

Figura 1: Primeiro caso teste

Na figura 2 representamos o segundo caso teste. Inicialmente, temos o rato numerado em 0. Após o primeiro comando, o rato apenas se desloca até a posição indicada pela seta vermelha. Já o segundo comando, rotaciona o rato de número 1 em 90° sentido horário, deslocando-o para a direta como mostrado na figura pelo rato de número 2. O terceiro comando faz o rato girar 90° novamente, em relação ao seu próprio eixo, fazendo-o apontar para baixo, chegando portanto na posição do rato número 3. O último comando faz o mesmo que os dois comandos anteriores, fazendo com que o rato desloque-se para a sua direita (apontando para a esquerda em relação ao eixo cartesiano) chegando a origem novamente.

O terceiro caso teste é exatamente igual ao segundo, apenas agrupando os três últimos comandos em um só, usando a estrutura de repetição da linguagem.

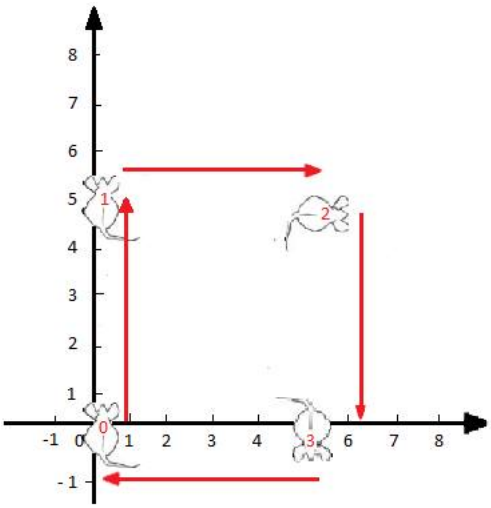

Figura 2: Segundo caso teste

Problema M. Yabba-Dabba-Doo!

Arquivo-fonte: `yabba.c`, `yabba.cpp` ou `yabba.java`

Fredi Flintistone era o maior jogador de boliche da idade da pedra! As regras daquela época eram as mesmas de hoje em dia. Deve-se conseguir a maior pontuação possível nas partidas. Para obter pontos, é preciso arremessar um côco sobre a pista para atingir 10 pinos de pedra, que estão dispostos em uma formação triangular.

O jogo do boliche consiste de 10 turnos. Em cada turno, o jogador Fredi lança a bola a fim de derrubar o maior número possível de pinos. Se todos os pinos forem derrubados, Fredi faz um Strike, registrado com um X no placar do jogo e seu turno se encerra. Caso contrário, o placar registra o número de pinos derrubados e Fredi faz um novo lançamento. Neste segundo lançamento, se derrubar todos os pinos restantes, Fredi faz um Spare, registrado por uma / (barra) no placar. Caso contrário, o número de pinos derrubados neste último lançamento é registrado.

A pontuação de um turno é o total de pinos derrubados, nos até dois lances mais a pontuação do turno anterior. Fredi recebe pontos extras caso tenha feito um Strike ou um Spare. Se fez um Strike, os pontos adquiridos nos dois lançamentos seguintes, mesmo que os lançamentos não sejam do mesmo turno, é somado ao do turno corrente, e se for um Spare, apenas os pontos do próximo lançamento é contabilizado como ponto extra. Caso Fredi faça um Strike ou um Spare no último turno, ele terá 2 ou 1 lançamentos extras, respectivamente. Os lançamentos extras são contabilizados **apenas** para calcular os pontos extras do último turno, ou seja, o último turno do jogo continua sendo o décimo.

Fredi possui o placar mas não possui os pontos contabilizados. Então ele pediu para você contabilizar os pontos do último turno para ele.

Entrada

A entrada contém os 10 turnos separados por espaço, e cada turno representado por um, dois ou três caracteres registrando o número de pinos derrubados.

Saída

Para cada caso, deve mostrar quantos pontos Fredi conseguiu.

Exemplos

Entrada	Saída
90 8/ 90 70 81 70 9/ 72 X 72	114

Entrada	Saída
90 07 80 80 9/ 90 07 03 9/ 80	96

Entrada	Saída
X X X X X X X X X XXX	300

No primeiro caso teste, temos as pontuações:
1º turno: 9 + 0 = 9 pontos
2º turno: 9 (pontuação anterior) + 10 (Spare) + 9 (extras) = 28
3º turno: 28 (pontuação anterior) + 9 + 0 = 37
4º turno: 37 (pontuação anterior) + 7 + 0 = 44
5º turno: 44 (pontuação anterior) + 8 + 1 = 53
6º turno: 53 (pontuação anterior) + 7 + 0 = 60
7º turno: 60 (pontuação anterior) + 10 (Spare) + 7 (extras) = 77
8º turno: 77 (pontuação anterior) + 7 + 2 = 86
9º turno: 86 (pontuação anterior) + 10 (Strike) + 7 + 2 (extras) = 105
10º turno: 105 (pontuação anterior) + 7 + 2 = 114
Portanto, resposta final é 114.