Maratona de Programação da SBC 2020

Sub-Regional Brasil do ICPC

14 de Novembro de 2020

Caderno de Problemas

Informações Gerais

Este caderno contém 15 problemas; as páginas estão numeradas de 1 a 25, não contando esta página de rosto. Verifique se o caderno está completo.

Este conjunto de problemas também está sendo utilizado simultaneamente nas seguintes competições: Segunda Fecha Gran Premio de México 2020, Primera Fecha Gran Premio de Centroamérica 2020 e Torneo Argentino de Programación 2020.

A) Sobre os nomes dos programas

1) Para soluções em C/C++ e Python, o nome do arquivo-fonte não é significativo, pode ser qualquer nome.
2) Se sua solução é em Java, ela deve ser chamada *codigo_de_problema*`.java` onde *codigo_de_problema* é a letra maiúscula que identifica o problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo.
3) Se sua solução é em Kotlin, ela deve ser chamada *codigo_de_problema*`.kt` onde *codigo_de_problema* é a letra maiúscula que identifica o problema. Lembre que em Kotlin o nome da classe principal deve ser igual ao nome do arquivo.

B) Sobre a entrada

1) A entrada de seu programa deve ser lida da *entrada padrão*.
2) A entrada é composta de um único caso de teste, descrito em um número de linhas que depende do problema.
3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

C) Sobre a saída

1) A saída de seu programa deve ser escrita na *saída padrão*.
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

Promoção:

Sociedade Brasileira de Computação

v1.0

Problema A

Álbum de Figurinhas

O álbum de figurinhas da Subregional Nlogoniana do ICPC 2020 já está disponível na Nlogônia! Programadores competitivos de todo o país estão comprando álbuns e colecionando figurinhas para celebrar a competição.

Este álbum é especial porque todas as figurinhas são iguais: elas contêm uma foto do troféu deste ano. Para completar o álbum, basta coletar figurinhas suficientes para preencher todos os espaços nele.

Você pode se perguntar: qual a graça de colecionar essas figurinhas? Para deixar as coisas interessantes, as figurinhas são vendidas em pacotes, cada um com um número aleatório de figurinhas. Os fãs celebram quando encontram muitas figurinhas em um pacote, zoam aqueles azarados que encontram poucas figurinhas, e se vangloriam por preencher seus álbuns com poucos pacotes.

Você acabou de adquirir o seu álbum e está pronto para começar a preenchê-lo! Mas antes de comprar os pacotes de figurinhas, você se perguntou: em média, quantos pacotes são necessários para completar um álbum?

Entrada

Há apenas uma linha de entrada contendo três inteiros, N, A e B, separados por um espaço, satisfazendo $1 \leq N \leq 10^6$, $0 \leq A \leq B \leq 10^6$ e $B > 0$, onde:

- N é o número de figurinhas necessárias para preencher o álbum;
- A é o número mínimo de figurinhas em um pacote;
- B é o número máximo de figurinhas em um pacote.

O número de figurinhas em cada pacote é um inteiro uniformemente distribuído no intervalo fechado $[A, B]$.

Saída

A saída consiste de apenas uma linha, que deve conter o número esperado de pacotes necessários para completar um álbum. O número será considerado correto se estiver dentro de um erro absoluto ou relativo de 10^{-5} da resposta correta.

Exemplo de entrada 1	**Exemplo de saída 1**
`40 0 2`	`40.33333`

Exemplo de entrada 2	**Exemplo de saída 2**
`100 1 10`	`18.72727`

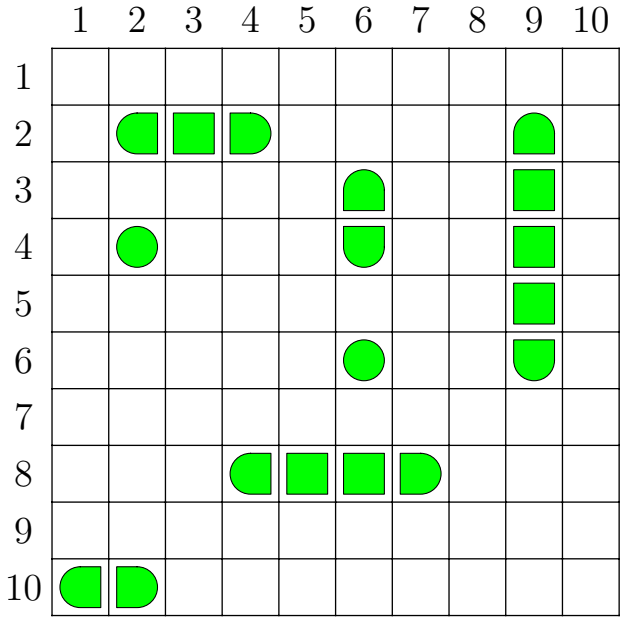
Exemplo de entrada 3	**Exemplo de saída 3**
`30 3 3`	`10.00000`

Exemplo de entrada 4	**Exemplo de saída 4**
`314 5 8`	`48.74556`

Problema B

Batalha Naval

Batalha Naval é um clássico jogo de estratégia para dois jogadores. Cada jogador posiciona seus navios num grid 10×10, e cada rodada do jogo consiste em adivinhar as posições dos navios do adversário. Existem muitas variações das regras, mas tais regras são irrelevantes para esse problema. Estamos interessados num problema mais básico: Dada a lista dos navios e suas posições, você deve determinar se o posicionamento inicial é válido.



As linhas e colunas do tabuleiro são numeradas de 1 a 10, e os navios são posicionados na horizontal ou na vertical, ocupando uma sequência contígua de quadrados do tabuleiro. Para esse problema, um posicionamento é válido se:

- nenhuma posição é ocupada por mais de um navio e;
- todos os navios estão inteiramente contidos no tabuleiro.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 100$), o número de navios. Cada uma das próximas N linhas contém quatro inteiros D, L, R e C com $D \in \{0, 1\}$, $1 \leq L \leq 5$ e $1 \leq R, C \leq 10$ descrevendo um navio. Se $D = 0$ então o navio está alinhado horizontalmente, e ocupa as posições $(R, C) \ldots (R, C + L - 1)$. Do contrário, o navio está alinhado verticalmente, e ocupa as posições $(R, C) \ldots (R + L - 1, C)$.

Saída

Imprima uma única linha contendo um único caractere. Se o posicionamento inicial dos navios for válido, então imprima o caractere maiúsculo '`Y`'; do contrário, imprima o caractere maiúsculo '`N`'.

Exemplo de entrada 1	**Exemplo de saída 1**
3 0 5 1 1 1 5 2 2 0 1 3 3	Y

Exemplo de entrada 2	**Exemplo de saída 2**
2 0 2 1 1 1 1 1 2	N

Exemplo de entrada 3	**Exemplo de saída 3**
1 0 2 10 10	N

Exemplo de entrada 4	**Exemplo de saída 4**
7 0 3 2 2 1 5 2 9 1 2 3 6 1 1 4 2 0 1 6 6 0 4 8 4 0 2 10 1	Y

Problema C

Concatenando Times

Pepito é um coach da Maratona que com frequência gosta de "concatenar" os nomes de dois times, tais como "AJI" e "Oxidados", a fim de obter nomes para novos times, tal como "AJIOxidados".

Dado que Pepito é coach de times de duas universidades onde ele leciona, ele teve uma ideia: ele vai considerar todas as possíveis concatenações de um nome de um time da universidade A, com um nome de um time da universidade B (sempre nesta ordem: primeiro o nome de um time da universidade A e depois o nome de um time da universidade B). Por exemplo, se os nomes dos times da universidade A são "Buen" e "Kilo", e se os nomes dos times da universidade B são "Pan" e "Flauta", as possíveis concatenações que ele considera são as cadeias "BuenPan", "BuenFlauta", "KiloPan" e "KiloFlauta".

Ele diz que um time é *peculiar* se a remoção desse time faz com que o conjunto de concatenações perca todas as concatenações que usam o nome desse time.

Pode-se verificar que no exemplo acima todos os times são peculiares. Contudo, se considerarmos o caso em que os nomes dos times de A são "xx" e "xxy", e os nomes dos times de B são "z", "yz" e "xx", então o time "xx" da universidade A não é peculiar, porque um dos nomes por ele gerado ("xx" + "yz" = "xxyz") pode ser também gerado sem usar o time em questão ("xxy" + "z" = "xxyz"). Pela mesma razão, "yz", "xxy" e "z" não são peculiares. O único time peculiar neste exemplo é "xx" da universidade B, porque é utilizado para criar os nomes "xxxx" e "xxyxx", e é absolutamente impossível criar qualquer um desses nomes sem usar "xx" da universidade B.

Dados os nomes dos times de ambas as universidades, sua tarefa é calcular quantos times peculiares existem em cada universidade.

Entrada

A primeira linha contém dois inteiros, M e N, separados por um espaço. O número de times da universidade A é M ($1 \leq M \leq 10^5$), e o número de times da universidade B é N ($1 \leq N \leq 10^5$).

A segunda linha contém os nomes dos M times da universidade A, separados por um espaço em branco e a terceira linha contém os nomes dos N times da universidade B, separados por um espaço em branco. Todos os nomes consistem apenas de letras minúsculas do alfabeto latino. Times distintos de uma mesma universidade têm nomes distintos.

A soma dos comprimentos dos nomes de todos os times é no máximo 10^6.

Saída

A saída deve conter apenas uma linha contendo dois inteiros: o número de times peculiares da universidade A e o número de times peculiares da universidade B, separados por um espaço em branco.

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

`2 2`
`buen kilo`
`pan flauta`
	`2 2`

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

`2 3`
`xx xxy`
`z yz xx`
	`0 1`

Problema D

Dança da Divisibilidade

No país da Nlogônia os habitantes realizam uma dança especial para homenagear o deus da divisibilidade. A dança é executada por N homens e N mulheres dispostos em dois círculos. Os homens ficam no círculo interno e as mulheres no círculo externo. Cada mulher inicia de frente para um homem.

A dança é composta de K movimentos; homens e mulheres se alternam nos movimentos, começando com os homens. No i-ésimo movimento, as pessoas do círculo correspondente rotacionam P_i passos em sentido horário enquanto as pessoas do outro círculo permanecem paradas. Assim, cada pessoa troca de parceiro para um que está a P_i posições de distância. Um movimento é válido se os parceiros de cada pessoa são diferentes ao início e ao fim do movimento e, além disso, nenhum par de pessoas está frente a frente em dois instantes de tempo distintos.

Como forma de homenagem, as danças sempre precisam terminar com casais cujas somas das idades tenham o mesmo resto quando dividido pelo número sagrado M. Ou seja, se a soma das idades de um casal deixa um resto R quando divido por M, então a soma das idades de todos os casais devem deixar o mesmo resto R ao fim da dança.

Fornecidos N, M e K e as idades de todos os dançarinos, determine de quantas formas se pode realizar a dança. Como a idade dos dançarinos é medida em segundos, os valores podem ser muito grandes.

Entrada

A primeira linha da entrada contém três inteiros N ($3 \leq N \leq 10^6$), M ($1 \leq M \leq 10^9$) e K ($1 \leq K \leq 10^9$), correspondendo à quantidade de pessoas em um círculo, ao número sagrado e à quantidade de movimentos da dança, respectivamente.

A segunda linha da entrada contém N inteiros A_i ($1 \leq A_i \leq 10^9$) separados por um espaço em branco e representando a idade das mulheres.

A terceira linha da entrada contém N inteiros B_i ($1 \leq B_i \leq 10^9$) separados por um espaço em branco e representando a idade dos homens.

Inicialmente o i-ésimo homem está alinhado com a i-ésima mulher, e o primeiro elemento de cada vetor é considerado à direita do respectivo último elemento.

Saída

A saída consiste de um único inteiro representando o resto da divisão do número de danças distintas por $10^9 + 7$.

Exemplo de entrada 1	**Exemplo de saída 1**
4 10 1 3 4 1 7 13 27 36 9	1

Exemplo de entrada 2	**Exemplo de saída 2**
5 10 2 3 4 1 7 6 4 7 1 2 5	3

Exemplo de entrada 3	**Exemplo de saída 3**
5 10 2 3 4 1 7 6 5 4 7 1 2	4

Exemplo de entrada 4	**Exemplo de saída 4**
6 21 3 10 58 23 31 37 2 45 17 9 24 38 30	42

Problema E

Empresa de Festas

Yankovich trabalha como Engenheiro de Software numa empresa, chamada POI, que promove festas online. Para testar os seus sistemas, os empregados organizaram festas e convidaram colegas, mas com algumas restrições.

A empresa tem uma estrutura hierárquica: Cada empregado, com exceção do dono da empresa, tem um gerente direto, e não há relações cíclicas de gerência. Devido ao processo de promoção da empresa, a idade de um empregado nunca é maior que a idade do seu gerente direto.

Serão organizadas M festas. A j-ésima festa tem um anfitrião e um intervalo de idades $[L_j, R_j]$. Para a j-ésima festa será convidado o maior conjunto de pessoas que satisfaça todas as restrições abaixo:

- O anfitrião participa da festa. Por isso, é garantido que a idade do anfitrião da j-ésima festa está no intervalo $[L_j, R_j]$.
- Todo convidado precisa ter idade no intervalo $[L_j, R_j]$.
- Todo convidado (que não o anfitrião) precisa trabalhar diretamente com (ou seja, ser gerente ou subordinado de) algum outro empregado que participa da festa.

Yankovich está responsável pelo programa que fornece informações sobre as festas das quais o usuário participou. Como uma tarefa inicial, ele tem que calcular de quantas festas cada empregado participou. Como ele está atrasado para entregar tal tarefa, ele pediu sua ajuda para escrever tal programa.

Entrada

A entrada consiste de várias linhas. A primeira linha contém dois inteiros N e M ($1 \leq N, M \leq 10^5$) representando o número de empregados e o número de festas de teste, respectivamente.

As próximas N linhas contêm a estrutura hierárquica da empresa. A i-ésima dessas linhas contém dois inteiros A_i e B_i ($1 \leq A_i \leq 10^5$, $1 \leq B_i \leq N$) representando a idade do i-ésimo empregado e seu gerente direto. Os empregados são numerados de 1 a N, com 1 representando o dono da empresa (ele é o único empregado com $B_i = i$). É garantido que $A_i \leq A_{B_i}$ para todo $1 \leq i \leq N$.

As próximas M linhas contêm os dados das festas de teste. A j-ésima dessas linhas contém três inteiros O_j, L_j, R_j ($1 \leq L_j \leq A_{O_j} \leq R_j \leq 10^5$) representando o anfitrião da festa e os limites do intervalo de idades descrito no enunciado.

Saída

Imprima uma única linha contendo N inteiros (separados por um único espaço). O i-ésimo desses números deve ser o número de festas de que o empregado i participou.

Exemplo de entrada 1	**Exemplo de saída 1**
10 3 8 1 3 5 5 1 2 3 4 1 3 3 1 2 7 1 2 2 3 2 3 5 9 5 3 8 3 2 6	2 1 3 1 1 2 0 2 0 1

Problema F

Fastminton

A Comissão RegionaL de Fastminton (CRLF) organiza torneiros anuais deste novo e inusitado esporte derivado do badminton. Neste ano, ocorrerá a terceira edição do grande torneio.

O antigo programador da comissão (sobrinho da diretora) desenvolveu um sistema para capturar e armazenar o resultado de cada ponto de uma partida, cujo resultado é salvo para um arquivo. Com a saída do antigo programador, que deixou para trás algumas versões defeituosas de seus programas, a CRLF precisa de você para garantir que os registros das emocionantes jogadas não sejam perdidos, confiando-lhe a tarefa de escrever um programa para ler os resultados dos arquivos de registro.

Para auxiliá-lo, a CRLF disponibilizou um resumo com as regras relevantes do Fastminton, que é, basicamente, uma versão mais curta (menor número de games) do badminton:

- As partidas de Fastminton são jogadas sempre com dois jogadores (oponentes) em uma quadra separada ao meio por uma rede;
- Os jogadores são identificados pela sua posição no placar (jogador da esquerda, jogador da direita);
- Uma partida é composta por três *games*. Ganha quem alcançar dois *games*;
- Ganha o *game* quem alcançar ao menos 5 pontos e tiver uma diferença de ao menos 2 pontos do oponente, ou o primeiro a chegar em 10 pontos;
- O jogador da esquerda inicia sacando no primeiro *game* da partida; nos demais, o jogador que inicia sacando é o que ganhou o último game;
- Cada jogada resulta em um ponto, de quem sacou ou de quem recebeu o saque. Quem ganhou o ponto irá sacar na próxima jogada.

Entrada

A entrada é composta por uma única linha contendo uma sequência de caracteres representando a sequência completa dos eventos de uma partida, podendo conter os caracteres `S` (ponto de quem sacou), `R` (ponto de quem recebeu o saque) ou `Q` (anúncio de placar). A entrada não contém anúncios de placar consecutivos.

Saída

O programa deverá imprimir uma linha contendo o placar atual para cada anúncio de placar (`Q`) encontrado.

Caso a partida esteja em andamento, o anúncio deverá ser na forma "`GL (PL) - GR (PR)`", onde `GL` e `GR` representam o número de *games* ganhos pelos jogadores da esquerda e da direita, e `PL` e `PR` são os pontos atuais dos jogadores da esquerda e da direita. Um asterisco (`*`) deverá ser adicionado no final do marcador de pontos do jogador que irá sacar na próxima jogada.

Caso a partida já esteja concluída, o anúncio será na forma "`GL - GR`" com a palavra "`(winner)`" adicionada no final do marcador de games do jogador ganhador da partida.

Exemplo de entrada 1	**Exemplo de saída 1**
`SRSSQSSSSQRRSS`	`0 (1) - 0 (3*)` `0 (0) - 1 (2*)`

Exemplo de entrada 2	**Exemplo de saída 2**
SRSSQSSSSQRRSSQ	0 (1) - 0 (3*) 0 (0) - 1 (2*) 0 - 2 (winner)

Exemplo de entrada 3	**Exemplo de saída 3**
RSRSSRRRRRRRRRSSSSRRSQ	2 (winner) - 0

Problema G

Game Show!

A Sociedade de Bons Competidores (SBC) organiza shows televisivos (e agora também transmitidos online!) para os seus competidores filiados. A SBC usa um sistema de créditos, denominados *sbecs*, que podem ser usados para participar de suas competições ou podem ser trocados por prêmios no final de cada temporada. Eles iniciaram um novo tipo de jogo, e precisam fazer algumas simulações para evitar prejuízos muito grandes na premiação!

Para participar deste novo jogo, o competidor precisa apostar 100 sbecs, que são transferidos para seu saldo no jogo, e uma sequência de caixas é posicionada. O jogo consiste de rodadas e o número máximo de rodadas é igual ao número de caixas. A cada rodada o jogador decide se abre a próxima caixa ou se encerra o jogo. Se ele encerrar, ele recebe seu saldo corrente de sbecs de volta. Se ele abrir a caixa, um número secreto, contido na caixa, é adicionado ao seu saldo e o jogo continua. Como o número secreto pode ser negativo, jogadores muito gananciosos podem acabar saindo no prejuízo! O jogo termina quando o jogador resolve encerrá-lo ou quando a última caixa é aberta.

A SBC contratou você para testar o jogo. A partir do conteúdo das caixas, você deve decidir qual seria a maior premiação possível que um jogador poderia conseguir.

Entrada

A primeira linha da entrada contém um inteiro C, $1 \leq C \leq 100$, o número de caixas do jogo. Depois, cada uma das C linhas seguintes descrevem, em ordem, o conteúdo das C caixas. Cada um delas contém um inteiro V, $-1000 \leq V \leq 1000$, correspondente ao conteúdo da caixa correspondente.

Saída

A saída consiste de uma única linha contendo um inteiro correspondente à maior premiação possível para aquela sequência de caixas.

Exemplo de entrada 1	**Exemplo de saída 1**
4 -1 -2 -3 -4	100

Exemplo de entrada 2	**Exemplo de saída 2**
5 -10 20 -30 40 -50	120

Problema H

Hangar do SBC

Um pequeno avião de carga do Sistema Binário de Cargas (SBC) foi projetado para transportar produtos especiais e secretos. Esses produtos são agrupados em caixas com diversos pesos.

O avião tem uma faixa de peso de segurança, dentro da qual a aeronave fica estável. Mais especificamente, existe um intervalo tal que se o peso total das caixas transportadas ficar fora desse intervalo então não será possível garantir a estabilidade do voo.

Sabe-se que todas as caixas têm pesos distintos. Além disso, dadas duas caixas, a mais pesada pesa pelo menos o dobro da caixa mais leve.

Sua tarefa é determinar de quantas formas se pode escolher um número especificado de caixas para se transportar no avião sem desestabilizá-lo.

Entrada

A primeira linha da entrada contém dois inteiros, N e K, que representam o número de caixas disponíveis e o número de caixas que devem ser embarcadas no avião, respectivamente.

A segunda linha da entrada contém N inteiros, separados por um espaço em branco, que representam os pesos das caixas.

A terceira linha da entrada contém dois inteiros, A e B, que especificam o intervalo de segurança dos pesos, que é o intervalo (fechado) $[A, B]$.

Considere todos os pesos informados na mesma unidade.

- $1 \leq N \leq 50$.
- $1 \leq K \leq 50$.
- o peso P de cada caixa está no intervalo $1 \leq P \leq 10^{18}$.
- $1 \leq A \leq B \leq 2 \times 10^{18}$.

Saída

A saída consiste de uma única linha, que contém o número de diferentes escolhas de caixas na quantidade especificada, sem por em risco o voo.

Exemplo de entrada 1	**Exemplo de saída 1**
3 2 10 1 3 4 13	3

Exemplo de entrada 2	**Exemplo de saída 2**
4 3 20 10 50 1 21 81	4

Exemplo de entrada 3	**Exemplo de saída 3**
6 3 14 70 3 1 6 31 10 74	11

Problema I

Interatividade

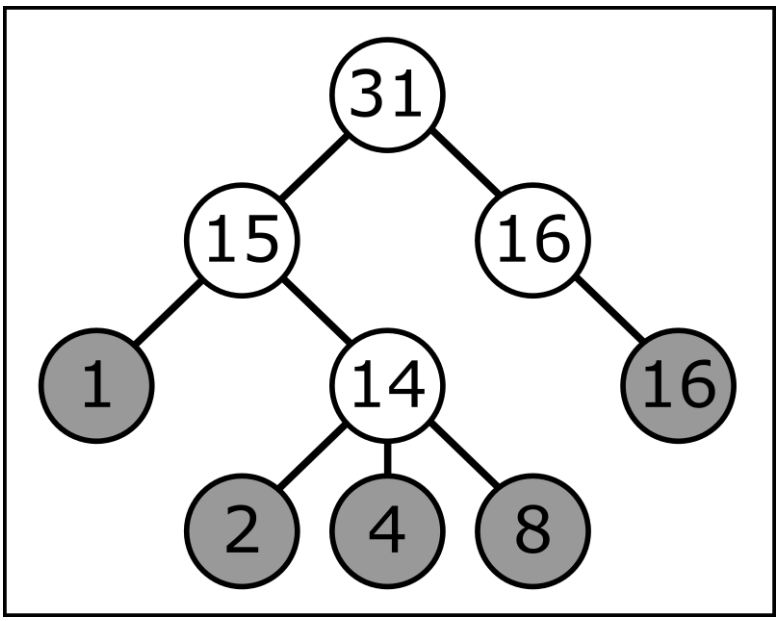
Um dia, Alice desafiou Beto com o problema interativo de programação descrito a seguir.

———————≀———————

Você tem uma árvore (um grafo acíclico conexo). Cada nó da árvore tem exatamente um *pai*, com exceção do nó *raiz*, que não tem pai. Um nó que não é pai de nenhum outro nó é uma *folha*. Você conhece a estrutura da árvore, porque sabe qual é o pai de cada nó que não é a raiz.

Cada nó contém um valor inteiro. Um nó que não é folha contém a soma dos valores dos seus filhos diretos. Portanto, todos os valores da árvore são determinados pelos valores contidos nas folhas.

A figura abaixo mostra um exemplo. As folhas estão marcadas como cinza, enquanto os outros nós são brancos. Cada nó mostra o valor contido nele.



Inicialmente, você não sabe o valor de nenhum nó da árvore, mas pode consultá-los um por um. Sua tarefa é determinar o valor de cada nó da árvore, usando o mínimo de consultas possível.

———————≀———————

Beto resolveu este problema facilmente. Então, para dificultar as coisas, Alice perguntou para ele: "dada a estrutura da árvore, quantas formas diferentes de solucionar este problema existem?" Isto é, quantos conjuntos mínimos de consultas existem que lhe permitam determinar os valores armazenados em cada nó da árvore? (Dois conjuntos de consultas são considerados diferentes se e somente se existe um nó consultado em apenas um dos dois conjuntos.)

Entrada

A árvore tem N nós no total. Cada nó é identificado por um inteiro entre 1 e N, onde o nó 1 é a raiz.

A entrada consiste de duas linhas. A primeira linha contém apenas o inteiro N.

A segunda linha contém $N-1$ inteiros $P_1, P_2, \ldots, P_{N-1}$, separados por um espaço, onde P_i é o pai do nó $i+1$, para $i = 1, 2, \ldots, N-1$.

$2 \leq N \leq 10^5$.

$1 \leq P_i \leq N$, para $i = 1, 2, \ldots, N-1$.

Saída

A saída consiste de uma única linha, que deve conter o número de soluções mínimas diferentes para o problema enfrentado por Beto. Como esse número pode ser muito grande, sua resposta deverá ser calculada módulo 1000000007 ($10^9 + 7$).

Exemplo de entrada 1	**Exemplo de saída 1**
3 1 1	3

Exemplo de entrada 2	**Exemplo de saída 2**
4 1 2 3	4

Exemplo de entrada 3	**Exemplo de saída 3**
5 1 2 2 2	7

Problema J

Juntando Dados

Acre e Amanda são muito curiosos e estão sempre procurando padrões à sua volta. Eles rotineiramente coletam e analisam dados de várias fontes (tráfego na cidade, volume de chuvas, número de folhas que caem das árvores), na esperança de encontrar padrões interessantes.

Na sua última expedição eles obtiveram um conjunto de dados bastante promissor: formava uma linha reta perfeita! Formalmente, era uma lista de $N/2$ pares de inteiros, possivelmente repetidos. Quando esses pares eram representados por pontos no plano cartesiano, todos os pontos eram perfeitamente colineares! Maravilhados, Acre e Amanda armazenaram estes dados como uma tabela contendo os pares de inteiros.

Infelizmente, enquanto Acre e Amanda saíram para coletar mais dados, seu filho pequeno entrou no escritório deles e bagunçou a tabela, trocando os valores de lugar. Agora só o que Acre e Amanda têm são os N valores da tabela, embaralhados. Eles querem tentar reconstruir a tabela original a partir deles.

Formalmente, Acre e Amanda querem agrupar esses números em $N/2$ pares, onde cada par representa um ponto no plano cartesiano, de tal forma que todos esses pontos sejam colineares. A lista de inteiros pode ter valores repetidos, e cada valor deve ser utilizado exatamente tantas vezes quantas aparece na lista. O conjunto de dados resultante pode ter pontos repetidos.

Acre e Amanda querem saber quantos conjuntos de dados diferentes podem ser formados com a lista de inteiros dada, já que podem haver vários. Dois conjuntos de dados são diferentes se e somente se existe um ponto que aparece mais vezes em um dos conjuntos do que no outro.

Entrada

A primeira linha contém um único inteiro N, o tamanho da lista de inteiros dada. N é sempre par, pois é o dobro da quantidade de pontos do conjunto de dados original. A segunda linha contém N inteiros, que representam a lista dos valores da tabela, embaralhados.

Os inteiros são separados por um único espaço.

$4 \leq N \leq 200$.

Cada número I da lista está no intervalo $-10000 \leq I \leq 10000$.

Saída

A saída deve consistir de uma única linha com um único inteiro, o número de diferentes maneiras de arranjar a lista de inteiros em pares que representem pontos colineares. Como esse número pode ser muito grande, sua resposta deverá ser calculada módulo 1000000007 ($10^9 + 7$).

A resposta poderá ser zero para alguns casos.

Exemplo de entrada 1	**Exemplo de saída 1**
8 1 2 3 5 20 18 16 12	2

Exemplo de entrada 2	**Exemplo de saída 2**
6 1 2 3 4 5 20	0

Exemplo de entrada 3	**Exemplo de saída 3**
8 1 2 5 5 5 5 8 9	10

Problema K

Ká entre Nós

Empates são sempre um problema em eleições ou em jogos. Recentemente, um novo jogo, chamado *Ká entre Nós*, foi inventado. O jogo é disputado por jogadores conectados numa rede social. Cada jogador tem um conjunto de amigos. A cada rodada há várias votações, mas um competidor somente pode receber votos de seus amigos. Ganha o jogador que receber o maior número de votos.

O jogo ainda está na fase de projeto, mas os desenvolvedores depararam com um problema muito comum. Dado que o número de amigos de cada jogador é em geral pequeno, empates são muito comuns, o que tira a graça do jogo. Para resolver esse problema, os desenvolvedores decidiram adicionar um novo módulo ao jogo. Esse módulo define os amigos de cada jogador, e sempre que possível dará a cada jogador um número ímpar de amigos.

O problema se mostrou mais complicado do que eles esperavam e agora estão tentando uma variação mais simples: dado um conjunto de jogadores, o módulo deverá obter uma *partição* dos jogadores em no máximo dois grupos, satisfazendo a restrição que cada jogador deve ter um número ímpar de amigos no seu grupo. Acontece que nem sempre isso é possível. Sua tarefa é decidir se é ou não possível obter a partição.

Entrada

A primeira linha da entrada contém dois inteiros, P e F, respectivamente o número de jogadores e o número de amizades, onde $2 \leq P \leq 100$ e $1 \leq F \leq P \times (P-1)/2$. Cada uma das próximas F linhas contém dois inteiros, A e B, indicando que A e B são amigos, onde $1 \leq A, B \leq P$ e $A \neq B$. Cada relação de amizade é dada no máximo uma vez, isto é, se uma linha contém os inteiros A e B, nenhuma outra linha contém tais inteiros.

Saída

A saída contém uma única linha, contendo um único caractere. Se for possível fazer a partição em dois grupos, escreva a letra maiúscula '`Y`'; caso contrário, escreva a letra maiúscula '`N`'.

Exemplo de entrada 1	**Exemplo de saída 1**
4 4 4 2 1 3 2 3 1 4	Y

Exemplo de entrada 2	**Exemplo de saída 2**
4 3 4 2 2 3 1 2	Y

Exemplo de entrada 3	**Exemplo de saída 3**
5 5 3 5 3 1 1 4 2 5 2 4	N

Problema L

Lavaspar

Caça Palavras é um passatempo bastante conhecido, embora esteja perdendo um pouco do seu prestígio nos últimos anos. O objetivo deste jogo é encontrar palavras em uma matriz, onde cada célula dessa matriz contém uma letra.

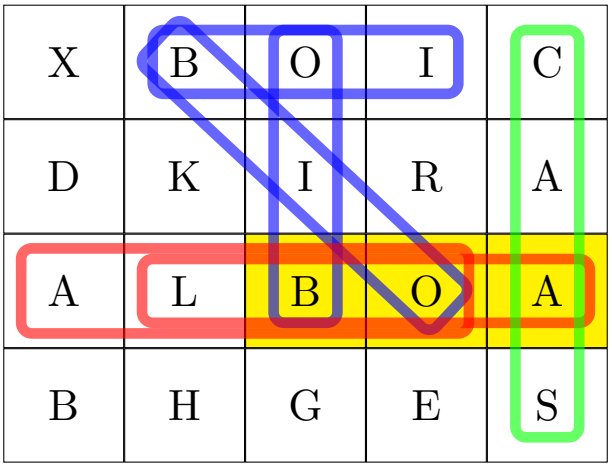
Bibika e seu irmão estavam jogando Caça Palavras, porém em pouco tempo perderam o interesse, visto que encontrar todas as palavras estava ficando relativamente fácil. Como Bibika queria que seu irmão saísse um pouco do computador, ela pesquisou na internet jogos do mesmo estilo e acabou encontrando o *Caça Lavaspar*.

Caça Lavaspar é um jogo que segue a mesma ideia do famoso Caça Palavras. Porém, ao invés de simplesmente ter que encontrar uma palavra na matriz, o objetivo é encontrar um anagrama qualquer da palavra, fazendo assim com que o jogo fique mais difícil e interessante. O anagrama pode ser encontrado em uma linha, coluna ou diagonal.

Um anagrama de uma palavra é formado pelo rearranjo das letras da palavra. Às vezes, o anagrama não tem sentido, mas isto não importa. BALO, LOBA e AOLB são exemplos de anagramas da palavra BOLA.

Bibika percebeu ser possível que uma mesma célula da matriz fizesse parte de anagramas de diferentes palavras e então ela passou a chamar essas células de *células especiais*.

Agora ela gostaria de saber, dada uma configuração de uma matriz e uma coleção de palavras, quantas células especiais existem?



A imagem acima ilustra o primeiro exemplo, onde a coleção de palavras consiste de três palavras: BOLA, CASA e BOI. Os retângulos de cada cor representam anagramas de palavras diferentes da entrada. As 3 células especiais estão pintadas de amarelo.

Entrada

A primeira linha possui dois inteiros L e C, que correspondem ao número de linhas e de colunas da matriz, respectivamente.

Seguem então L linhas, cada uma contendo uma palavra com C letras.

Após isso, a próxima linha contém um inteiro, N, que representa a quantidade de palavras na coleção de palavras a seguir.

E então, por fim, temos mais N linhas, onde cada uma delas contém uma palavra da coleção.

Todos os caracteres utilizados, tanto na matriz quanto na coleção de palavras, são letras maiúsculas do alfabeto inglês.

É garantido que nenhum par de palavras da coleção é um anagrama uma da outra.

- $2 \leq L, C \leq 40$.
- $2 \leq N \leq 20$.
- O número P de letras de cada uma das N palavras está no intervalo $2 \leq P \leq \min(15, \max(L, C))$.

Saída

A saída deve consistir de uma única linha que contém o número de células especiais.

Exemplo de entrada 1	**Exemplo de saída 1**
4 5 XBOIC DKIRA ALBOA BHGES 3 BOLA CASA BOI	3

Exemplo de entrada 2	**Exemplo de saída 2**
3 3 AAB ABA BAA 2 ABA BBB	3

Exemplo de entrada 3	**Exemplo de saída 3**
2 4 AAAA AAAA 2 AAA BBB	0

Problema M

Metralhadora

Fulanito foi jogar um arcade das antigas. No jogo, ele pode colocar uma metralhadora em qualquer lugar da sua base, que consiste de todos os pontos (x, y) com coordenadas inteiras e $x < 0$. Há N inimigos no campo de batalha. O i-ésimo inimigo $(1 \leq i \leq N)$ está na posição (x_i, y_i) com $x_i > 0$. Todas as posições são dadas de antemão.

Uma metralhadora posicionada em (x_m, y_m) cobre um ângulo de visão para a direita centrado na reta $y = y_m$, cujos limites são dados pelas retas $y = y_m \pm \frac{x - x_m}{2}$. Quando colocada, ela atinge todos os inimigos na região delimitada por esse ângulo, incluindo os localizados nas retas-limite.

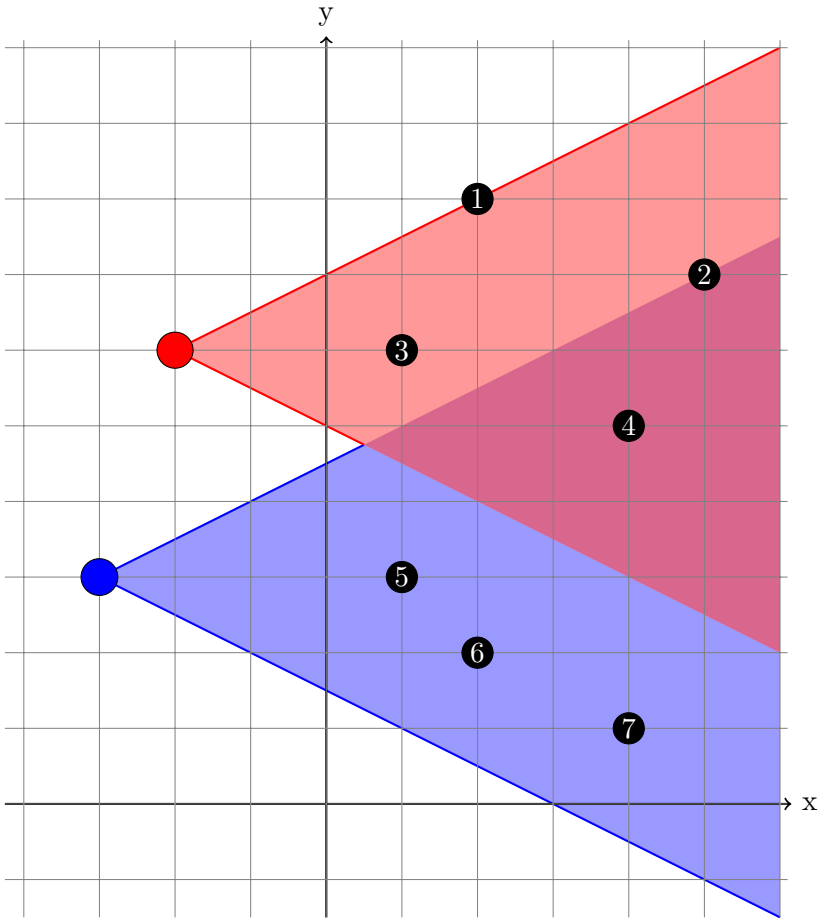


Figura 1: Representação pictória da entrada de exemplo

O sistema de pontuação usado por esse jogo é desnecessariamente complicado; muitos acreditam que tal sistema foi um grande erro dos desenvolvedores (que, em resposta, afirmam com convicção que "não é um bug, é um recurso!"). Especificamente, a pontuação obtida por um dado posicionamento da metralhadora é calculada executando os seguintes passos:

- Liste os índices (i entre 1 e N) de **todos** os inimigos que a metralhadora atinge.
- Ordene os índices em ordem crescente, e chame os valores ordenados de $i_0 < i_1 < \cdots < i_{k-1}$.
- Compute a pontuação usando a fórmula $\left(\sum_{j=0}^{k-1} i_j \cdot 5782344^j\right) \bmod (10^9 + 7)$, onde $a \bmod b$ denota o resto da divisão de a por b.

- Nota: Uma metralhadora que não atinge inimigos recebe uma pontuação exatamente igual a 0.

Para melhorar nesse jogo, Fulanito te faz Q perguntas: cada consulta pede o placar que seria obtido se posicionássemos a metralhadora numa certa posição (x_m, y_m). Para tornar o problema mais desafiador, os valores de (x_m, y_m) não são dados diretamente. Ao invés disso, são dados valores a e b que podem ser usados para calcular x_m e y_m através das fórmulas $x_m = -1 - \big((p + a) \bmod (10^9 + 7)\big)$ e $y_m = (p + b) \bmod (10^9 + 7)$, onde p é a resposta da consulta anterior ($p = 0$ ao processar a primeira consulta).

NOTA: É garantido que a soma do número de inimigos atingidos em todas as consultas é no máximo 10^6.

Entrada

A entrada consiste de várias linhas. A primeira linha da entrada contém dois inteiros N, Q ($1 \leq N, Q \leq 10^5$), o número de inimigos e o número de consultas.

As próximas N linhas da entrada contém dois inteiros cada: x_i e y_i ($1 \leq x_i, y_i \leq 10^9$), as coordenadas da posição do i-ésimo inimigo.

As próximas Q linhas contém dois inteiros cada: Os valores a e b ($0 \leq a, b < 10^9+7$) que especificam cada consulta, como explicado no enunciado.

Saída

Para cada consulta, imprima um único inteiro contendo a resposta para a consulta.

Exemplo de entrada 1	**Exemplo de saída 1**
7 2 2 8 5 7 1 6 4 5 1 3 2 2 4 1 2 3 373785639 373785644	626214369 981053491

Problema N

Números Multiplicados

Eugênio é um brilhante matemático que se diverte multiplicando números.

Certa vez, ele encontrou M pedaços de papel, numerados de 1 a M, cada um com um vértice desenhado. Chamaremos tais vértices de *M-vértices*. Cada um desses vértices estava rotulado com um primo distinto. Além disso, os primos estavam ordenados: Se chamarmos o rótulo do vértice no i-ésimo pedaço de papel de p_i, então $p_i < p_j$ para todo par $i < j$.

Após encontrar os pedaços de papel, Eugênio decidiu desenhar N outros vértices, que chamaremos de *N-vértices*, e adicionar arestas entre os M-vértices e os N-vértices. Ele tomou o cuidado de nunca ligar um M-vértice com um M-vértice, nem um N-vértice com um N-vértice, mas não se preocupou com o número de arestas desenhadas entre dois vértices. Assim, ele obteve um multigrafo bipartido.

Como o principal interesse de Eugênio é multiplicar números, ele decidiu rotular cada N-vértice com a multiplicação de todos os M-vértices conectados a ele. Se um M-vértice estiver conectado a um N-vértice por várias arestas, o rótulo dele será multiplicado várias vezes (igual ao número de arestas que os conecta) no processo de formar o rótulo do N-vértice.

Cada N-vértice i acabou rotulado com um número c_i. Formalmente, podemos escrever a seguinte fórmula para c_i:

$$c_i = \prod_{(j,i) \in E} p_j,$$

onde E é o multiconjunto de arestas (cada elemento de E é um par da forma (m, n) com $1 \leq m \leq M$ e $1 \leq n \leq N$). Depois de construir os rótulos dos N-vértices, Eugênio foi comprar um lanche, que consistiu de um toro e um café. Ao saborear o toro, Eugênio acidentalmente derramou o seu café, tornando os rótulos $p_1, \ldots, p_M$ dos M-vértices ilegíveis.

Você pode ajudá-lo a recuperar os números primos ordenados destruídos pelo café?

Entrada

A primeira linha contém três inteiros M, N e K, o número de M-vértices, o número de N-vértices e o número de arestas *distintas*. Tais valores satisfazem $1 \leq M, N < 10^3$ e $1 \leq K < 10^4$.

A próxima linha contém N números c_i, os rótulos dos N-vértices. Tais valores satisfazem $1 < c_i < 10^{15}$.

Finalmente, há K linhas, cada uma contendo três números m, n e d, representando que há d arestas entre o M-vértice m e o N-vértice n. Tais números satisfazem $1 \leq m \leq M$, $1 \leq n \leq N$ e $1 \leq d \leq 50$.

É garantido que todos os vértices (tanto M-vértices quanto N-vértices) têm grau pelo menos um. Em outras palavras, todo vértice tem pelo menos uma aresta conectada a ele.

Saída

Imprima uma única linha com M números ordenados, os primos rótulos dos M-vértices de índices $1, \ldots, M$ que fizeram Eugênio perder o sono.

Exemplo de entrada 1	**Exemplo de saída 1**
4 3 4 15 16 13 2 1 1 3 1 1 1 2 4 4 3 1	2 3 5 13

Exemplo de entrada 2	**Exemplo de saída 2**
4 5 7 3 9 7 143 143 1 1 1 1 2 2 2 3 1 3 4 1 3 5 1 4 5 1 4 4 1	3 7 11 13

Problema O

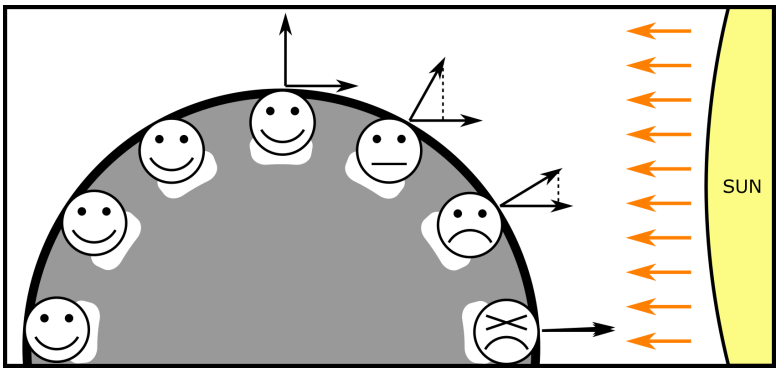
Ônibus Venusiano

A Colônia Humana em Vênus está prosperando! Aqui, o meio de transporte mais usado é o Ônibus Venusiano: um disco voador com janelas e assentos ao longo de suas bordas. Nesse ônibus, todos os assentos são nas janelas. E não é permitido mudar de assento. Portanto, uma vez que uma pessoa escolhe um lugar, ela deve permanecer nele até descer do ônibus.

Apesar de ser um veículo completamente autônomo, cada ônibus opera com um engenheiro a bordo, para lidar com problemas inesperados. Você é o engenheiro do ônibus 1C9C, e passa a maior parte do seu expediente lendo livros. O problema é que você detesta ficar ao sol. Portanto, você quer escolher um lugar pra sentar que minimize o total de luz solar que você vai receber ao longo do seu expediente de trabalho.

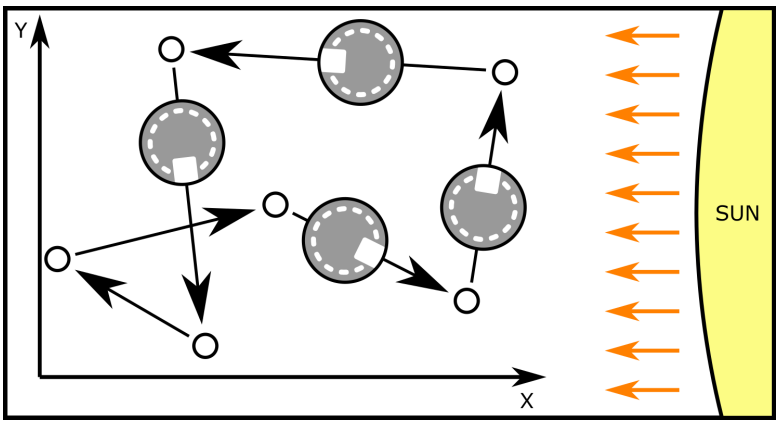
A colônia é representada pelo plano cartesiano, onde o eixo X aponta para o leste e o eixo Y aponta para o norte. Os dias em Vênus são bem longos (mais longos até do que o ano!), então você pode assumir que o sol sempre vem da direção leste. Isto é, a luz solar sempre viaja para o oeste, na direção contrária ao eixo X.

Veja a figura abaixo. Quanto mais sua janela estiver virada para o leste, mais luz solar você tem que aguentar. Mas se sua janela estiver virada para o oeste, você não recebe nenhum sol.



Formalmente, suponha que o vetor (D_x, D_y) representa a direção para a qual a sua janela está virada. Note que você só recebe sol se $D_x > 0$. E seja θ o ângulo entre os vetores (D_x, D_y) e $(1, 0)$ (um vetor apontando diretamente para o sol). Se $\cos(\theta) \leq 0$, você não recebe nenhum sol. Caso contrário, você recebe $\cos(\theta)$ unidades de luz solar por segundo.

A rota do ônibus consiste de uma sequência de estações ao redor da colônia. O ônibus começa o expediente na primeira estação, visita todas as estações em ordem, e então retorna à primeira.



O trajeto entre duas estações consecutivas é sempre em linha reta, com velocidade constante de

um metro por segundo. E apesar do ônibus ser redondo, ele tem um "lado da frente": este lado está sempre virado para a direção que o ônibus se move, e o ônibus gira apropriadamente quando muda de direção nas estações.

Você pode ignorar o tempo que o ônibus gasta mudando de direção, coletando ou largando passageiros.

Entrada

A primeira linha contém um único inteiro N, a quantidade de estações visitadas pela rota do ônibus.

Em seguida há N linhas, cada linha contendo as coordenadas X e Y de uma estação, separadas por um espaço.

As estações são dadas na ordem em que são visitadas.

Qualquer estação pode ser visitada mais de uma vez na rota.

Quaisquer duas estações consecutivas são distintas, assim como a última e a primeira estações.

Todas as coordenadas são dadas em metros.

$2 \leq N \leq 100000$.

As coordenadas de cada estação são inteiros no intervalo $-10000 \leq X, Y \leq 10000$.

Saída

A saída consiste de uma única linha que deve conter um número real, a quantidade mínima total de luz solar que você pode receber numa única jornada ao longo da rota do ônibus. Sua resposta deve ter exatamente duas casas decimais.

Exemplo de entrada 1	**Exemplo de saída 1**
3 2 5 17 5 11 11	6.00

Exemplo de entrada 2	**Exemplo de saída 2**
4 3 0 3 6 6 3 0 3	4.24

Exemplo de entrada 3	**Exemplo de saída 3**
4 3 2 1 1 -3 -1 -1 0	0.00