

A. Salvador

time limit per test: 1 second
memory limit per test: 256 megabytes



As sábias tartarugas: Rafael, Leonardo, Donatello e Michelangelo.

Na final latino-americana da ICPC 2025 em Salvador, a organização visitou o Projeto Tamar, onde diversas tartarugas marinhas são resgatadas e cuidadas. Entre elas, quatro são especialmente respeitadas por sua sabedoria e intelecto: Rafael, Leonardo, Donatello e Michelangelo.

A organização pediu que cada uma dessas tartarugas criasse um plano sobre como elas acham que a programação do evento deveria ser. Em seguida, foi realizada uma votação para escolher o plano vencedor. Cada pessoa pôde votar em apenas uma tartaruga.

Uma tartaruga vence se tiver mais da metade dos votos totais.

Até o momento, várias pessoas já votaram, mas N pessoas ainda não votaram, por estarem atrasadas. As tartarugas então se perguntam: quais delas ainda têm chance de vencer? Afinal, sabias como são, se a tartaruga não tem chance de ganhar ela prefere voltar para o mar e comer algas.

Sua tarefa é, dado o número atual de votos de cada tartaruga e o número de votos restantes, imprimir quais tartarugas ainda podem vencer a votação.

Input

A entrada consiste de uma linha com 5 inteiros v_R v_L v_D v_M N onde:

- v_R : votos atuais de Rafael
- v_L : votos atuais de Leonardo
- v_D : votos atuais de Donatello
- v_M : votos atuais de Michelangelo
- N : número de pessoas que ainda não votaram

É garantido que todos os inteiros estão entre 0 e 1 milhão.

Output

Imprima, em ordem alfabética, os nomes das tartarugas que ainda podem vencer a eleição (uma por linha). Se nenhuma tartaruga puder mais alcançar a maioria dos votos, imprima: "sem vencedores".

Examples

input
0 0 0 0 10000
output
Donatello Leonardo Michelangelo Rafael

input
5 4 3 3 0
output
sem vencedores

input
10 8 6 4 12
output
Rafael

input
8019 5386 7532 4929 6349
output
sem vencedores

B. Astana

time limit per test: 1 second
memory limit per test: 256 megabytes



Ber após ter ganho a competição de flexões. Para isso ele fez mais de 250 flexões, com pausas de 30 segundos a cada 50.

Em Astana, no Cazaquistão, em 2024, aconteceu a 48ª edição da ICPC (International Collegiate Programming Contest). E, junto com ela, foi realizada a primeira maratona de flexões da história da competição. O grande campeão foi Bernardo Archegas, da UNICAMP.

Como todos sabem, vencer uma maratona de flexões não é nada fácil. Para alcançar esse feito, Bernardo precisou treinar com muita disciplina. Para isso, ele utilizava o aplicativo GymRats, que funciona da seguinte maneira: todos os dias em que ele vai à academia e faz flexões, ele registra uma visita no app. Além disso, o aplicativo mostra qual foi a maior sequência de dias consecutivos em que ele foi para a academia sem falhar.

A sua tarefa é implementar essa funcionalidade do aplicativo. Dada a lista de dias (em ordem crescente) em que Bernardo foi para a academia, determine o tamanho da maior sequência de dias consecutivos em que ele compareceu.

Input

A primeira linha contém um número inteiro N ($1 \leq N \leq 1000$), representando a quantidade de dias em que Bernardo foi para a academia.

A segunda linha contém N inteiros distintos $1 \leq d_1 < d_2 < \dots < d_N \leq 1000$, representando os dias em que ele foi para a academia, em ordem crescente. Os dias são numerados sendo 1 o primeiro dia que Bernardo baixou o aplicativo e dia d_N o dia da competição.

Output

Imprima um único inteiro representando o tamanho da maior sequência de dias consecutivos em que Bernardo foi à academia.

Examples

input
6 2 5 6 7 10 11
output
3

input
5 10 100 101 102 103
output
4

C. João Pessoa

time limit per test: 1 second

memory limit per test: 256 megabytes



Equipes da UNICAMP na praia de João Pessoa, na final brasileira de 2024

João Pessoa, a capital da Paraíba, é conhecida por ser a primeira cidade do Brasil a ver o sol nascer. Em uma manhã especial na Praia do Jacaré, os moradores estavam se preparando para o famoso espetáculo do nascer do sol ao som do Bolero de Ravel. Mas, havia um problema curioso...

Um artista local estava preparando uma faixa decorativa feita apenas com parênteses '(' e ')', representando o movimento do sol no horizonte. A faixa deveria ser visualmente harmoniosa, ou seja, a sequência de parênteses precisava estar balanceada. Formalmente, para toda posição da faixa, o número de parênteses que se abrem até ali

nunca pode ser menor do que o número de parênteses que se fecham. Alguns exemplos são "()", "(()())" ou "(()())()()".

No entanto, na hora de colocar os parênteses na faixa ele reparou que colocou na ordem errada! Como o número total de '(' e ')' ainda está correto e igual, o artista decidiu fazer um recorte na faixa, tirando algumas parênteses do início e depois costurando no final, sem mudar a ordem ou sentido. Mais formalmente, ele faz uma rotação cíclica — ou seja, mover alguns caracteres do começo da faixa para o fim.

Sua tarefa é ajudar o artista a descobrir quantos caracteres do início precisam ser movidos para o final para que a sequência se torne válida. **É possível provar que nessas condições sempre existe uma rotação válida.** Se houver mais de uma maneira de fazer isso, você pode escolher qualquer uma delas.

Input

A primeira linha contém um número inteiro N ($1 \leq N \leq 5 \cdot 10^5$), representando o comprimento da faixa.

A segunda linha contém uma string S de tamanho N , composta apenas pelos caracteres '(' e ')'. É garantido que o número de '(' será igual ao número de ')'.

Output

Imprima um único inteiro $0 \leq r < N$, indicando que ao mover os primeiros r caracteres para o final teremos uma faixa válida. Ou seja, $S_{r+1}, S_{r+2} \dots S_N, S_1, S_2 \dots S_r$ é uma faixa válida. Se houver mais de uma resposta, imprima qualquer uma delas.

Examples

input
6)) ((()
output
2
input
2 ()
output
0
input
4)) ((
output
2

D. Luxor

time limit per test: 2 seconds
memory limit per test: 1024 megabytes



Leal com seu lenço árabe que negociou de 10 dólares para 1 dólar

Em Abril de 2024, ocorreu em Luxor, no Egito, duas finais mundiais, devido a um atraso do Covid. O Egito é conhecido por muitas coisas, como sua rica história, pirâmides deslumbrantes e por uma cultura de barganha intensa. É comum entrar em uma loja sem nenhum preço e ao perguntar quanto é cada coisa receber como resposta "não se preocupe Habibi, escolha tudo que quer e depois vemos o preço". Por conta disso, os preços dos itens das lojas variam muito, podendo alcançar dezenas de dólares ou apenas alguns centavos para outros.

Após suas aventuras no Egito, Matheus Leal decide abrir sua própria loja. Para isso faz uma compra grande de diversos itens, como chaveiros, pedras, esculturas, papiro, e outros souvenirs. Depois vem a parte divertida: a barganha com os clientes, uma luta de intelecto e força de vontade para ver quem consegue o melhor preço.

Porém, no final de todo mês vem a parte chata, fazer as contas. Durante o mês Leal anota todos os valores de compra e venda e tem um array A de tamanho N , com valores em centavos. Como bom cientista da computação, Leal programou o seu computador para somar os valores na ordem dada. Porém, para sua surpresa, nem sempre seu programa funciona!

O motivo é que a máquina de Leal tem apenas $(x + 1)$ -bits, ou seja, durante o seu algoritmo de soma, o valor nunca pode ser maior que $2^x - 1$ ou menor que -2^x , caso contrário ele terá overflow (ou underflow) e o resultado será errado!

A sua tarefa é ajudar Leal, reordenando a lista de forma que a soma nunca passe dos valores limites. Caso seja impossível, avise Leal!

Input

A primeira linha possui um inteiro N , $1 \leq N \leq 10^5$. A segunda linha possui um inteiro L , que é garantidamente uma potência de 2, indicando o valor de 2^x . É garantido também que $1 \leq L \leq 2^{20}$.

Segue uma linha com N inteiros separados por espaços, representando o array A . É garantido que $-L \leq A_i \leq L - 1$.

Output

Caso seja impossível, imprima apenas uma linha: "N".

Caso seja possível, imprima na primeira linha "S". Imprima então N inteiros separados por espaço, o array A após ser reordenado.

Examples

input
4
8
7 7 -8 1

output
S -8 1 7 7

input
4 8 7 7 -8 3
output
N

input
6 64 1 2 4 8 16 32
output
S 32 16 8 4 2 1

Note

Cuidado para você também não tomar overflow!! Segue um exemplo de como Leal computa sua receita:

```
receita = 0
for valor in a:
    receita += valor
```

E. Guadalajara

time limit per test: 1 second
memory limit per test: 256 megabytes



Essa é uma história **100%** verídica, que ocorreu em 2024, na primeira final latino americana.

Emanuel, do time "teambrbr002", foi para o aeroporto de Belo Horizonte com 10 horas de antecedência, onde pegaria um voo para a cidade de Guadalajara, no México, para a primeira competição "Programadores das América". Após muita conversa, resolução de problemas no terminal, e uma viagem para Guarulhos, Emanuel se dirige para a parte de imigração, quando se depara com um cena terrível. Algum ser do mal havia trocado seu passaporte por uma carteira de trabalho, ambos itens muito similares, e agora já é tarde demais!

Emanuel consegue convencer a equipe do aeroporto para ter uma nova passagem para o México, mas seria uma longa e solitária jornada fazendo o percurso: SP -> MG -> casa para pegar o passaporte -> SP -> Chile -> Panamá -> México.

Sem internet e sem seus amigos para conversar, Emanuel só tem agora consigo algumas moedas. Ele decide então jogar um jogo milenar, passado de geração em geração, que envolve apenas algumas moedas — mas que pode entreter (ou aprisionar) um jogador por inúmeras horas.

Inicialmente, as moedas são colocadas em uma fileira, algumas com a cara (representado pela letra H, do inglês Head) para cima, outras com a coroa (representado por um T, tails). Em cada jogada, o jogador deve:

- Escolher uma moeda qualquer que esteja com a cara (H) para cima e virá-la (ficando com T);
- Para cada moeda à esquerda da escolhida, ele pode decidir individualmente se deseja virá-la ou não.

Após cada jogada, o jogador repete o processo com a nova configuração de moedas. O jogo termina quando todas as moedas estão com a coroa (T) voltada para cima, pois não há mais jogadas possíveis.

Mas há uma lenda que diz que, dependendo da configuração inicial, o jogo pode durar para sempre, sem nunca chegar a um estado final.

Sua tarefa é escrever um programa que:

- Detecte se o jogo pode ser jogado infinitamente. Nesse caso, imprima apenas -1.
- Caso contrário, imprima a maior sequência possível de configurações das moedas após cada jogada válida, incluindo o estado inicial e o estado final. Caso existam múltiplas sequências de tamanho máximo imprima qualquer uma.

Cada configuração deve ser representada por uma string de caracteres, contendo apenas 'H' para cara (Heads) e 'T' para coroa (Tails), em letras maiúsculas.

Input

Uma única linha com uma string de até 15 caracteres, representando o estado inicial das moedas. Cada caractere será 'H' ou 'T'.

Output

Se for possível jogar para sempre, imprima apenas −1. Caso contrário, primeiro imprima uma linha com um inteiro *k*, o número de elementos da sequência de tamanho máximo. Após, imprima uma linha para cada configuração da sequência, começando pelo estado inicial e terminando no estado com todas as moedas viradas para Tails ('T'), uma configuração por linha. Caso existam múltiplas sequências de tamanho máximo imprima qualquer uma.

Examples

input
HH
output
4 HH TH HT TT

input
HTT
output
2 HTT TTT

input
TTT
output

Note

Após uma jornada de 3 dias, Emanuel e seu time foram os campeões da competição, com uma diferença de apenas 1 minuto do segundo lugar.

F. Chapecó

time limit per test: 1 second
memory limit per test: 256 megabytes



O time da UNICAMP ganhou o 1 lugar na maratona em Chapecó, feito conquistado apenas 3 outras vezes até o momento. De fato foi um momento muito feliz para todos da UNICAMP.

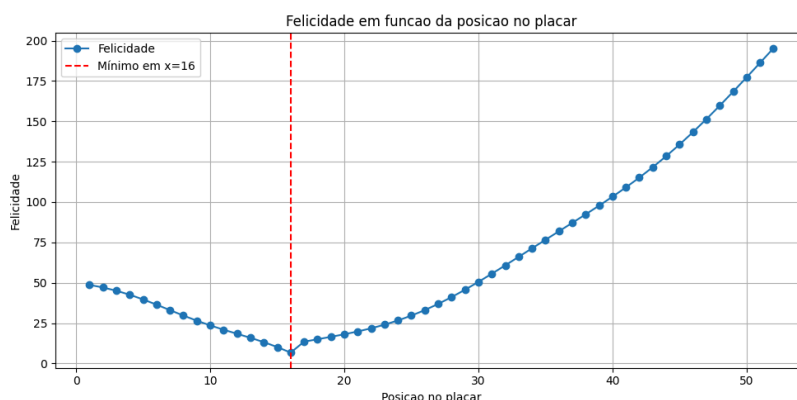
A final brasileira de 2023 aconteceu em Chapecó, sendo esta a primeira edição a classificar times para a final latino americana.

Durante a competição, algo trágico aconteceu com o time "Estamos Vivendo um Sonho": eles terminaram na 16ª colocação, mas apenas os 15 primeiros se classificaram para a Latam.

Comovida com a situação, a SBC decidiu aplicar um questionário para medir a felicidade de cada time após a competição, atribuindo a cada um uma nota de felicidade entre 0 e 10^9 . Após analisar os dados, uma curiosa conclusão surgiu: a felicidade dos times, em função da colocação final, se comporta como uma função não-crescente da posição 1 até a primeira posição a não se classificar para a latam, e como uma função não-decrescente da primeira posição a não se classificar até a última colocação.

Para evitar novas frustrações, os times da Unicamp pediram ao Carlinhos que revelasse o número de vagas disponíveis para a Latam. Porém, Carlinhos explicou que essa informação é confidencial. Ainda assim, usando seus misteriosos métodos estatísticos, ele aceitou responder perguntas informando o nível de felicidade de um time em uma determinada colocação.

Com essas respostas, os times perceberam que é possível determinar um intervalo contendo todos os possíveis valores para o número de vagas — mas precisam da sua ajuda para calcular esse intervalo.



Exemplo da felicidade em função da posição. Nos casos de testes as coordenadas são inteiras. Note que o formato da curva pode ser muito variado, mas é garantido que ela desce e depois sobe, possivelmente mantendo o mesmo valor por vezes.

Input

A primeira linha contém dois inteiros N e M ($1 \leq N \leq 10^9, 1 \leq M \leq 10^5$) — o número total de times competindo e o número de perguntas feitas a Carlinhos. As próximas M linhas contêm dois inteiros P e V ($1 \leq P \leq N, 1 \leq V \leq 10^9$) — indicando que a felicidade na colocação P é V .

É garantido que os valores de P são diferentes.

Output

Após cada uma das M perguntas, imprima dois valores L e R , representando o menor intervalo possível $[L, R]$ que contém todos os valores ainda possíveis para o número de vagas da Latam, com base nas perguntas feitas até aquele momento.

Note que é possível que o número de vagas seja 0.

Examples

input
20 5 16 4 14 10 18 15 19 16 20 18
output
0 20 14 20 14 16 14 16 14 16

input
10 4 2 5 4 5 6 5 8 5
output
0 10 0 10 0 10 0 10

input
10 5 8 4 5 2 6 1 7 3 9 5
output
0 10 0 6 5 6 5 5 5 5

Note

É possível que muitas posições tenham o mesmo valor de felicidade, incluindo a valor de felicidade mínimo.

G. Dhaka

time limit per test: 1 second
memory limit per test: 1024 megabytes



Tuk-tuks fazendo manobras radicais.

Nas movimentadas ruas de Dhaka, capital de Bangladesh, é comum ver os pequenos veículos chamados de **tuk-tuks**. Esses veículos de três rodas são uma forma popular e ágil de transporte urbano, bastante utilizados tanto por moradores locais quanto por turistas. Foi nessa cidade que ocorreu a final mundial da maratona, em novembro de 2022.

Todos os anos, Dhaka sedia uma corrida tradicional de tuk-tuks, que atrai a atenção de toda a cidade. Diferente de corridas convencionais com voltas e turnos, essa competição é bem caótica e divertida: a cada momento, qualquer tuk-tuk pode realizar uma manobra ousada, que lhe rende pontos e pode melhorar sua posição no ranking.

Você foi chamado para trabalhar com a equipe de juizes da competição e ficou responsável por acompanhar as posições dos tuk-tuks ao longo da prova, anotando toda vez que um tuk-tuk avançava de posição. Infelizmente, um dos assistentes esqueceu de registrar os valores de pontuação dos tuk-tuks — tanto os iniciais quanto os pontos ganhos em cada manobra.

Agora que a corrida acabou, os jornais estão exigindo os dados completos, e você precisa, com urgência, **reconstruir uma possível sequência de pontuações** compatível com os registros de posições.

Nessa tarefa, você receberá dois inteiros N e Q :

- N é o número de tuk-tuks participantes. Inicialmente, o tuk-tuk i está na posição i (sendo a posição 1 a melhor colocação).
- Em seguida, você receberá Q atualizações. Cada atualização informa que o tuk-tuk P passou a ocupar a posição X , que é melhor ou igual à posição anterior de P .

Sua tarefa é:

- Encontrar uma possível atribuição de pontuações iniciais aos N tuk-tuks;
- Para cada atualização, informar quantos pontos o tuk-tuk P ganhou na manobra que o levou à nova posição;

Você deve garantir que o ranking fique consistente após cada manobra — ou seja, os tuk-tuks devem permanecer ordenados por pontuação após cada atualização.

Todos os valores de pontuação (iniciais e ganhos) devem estar no intervalo $[0, 10^9]$. Além disso, as pontuações finais também devem estar nesse intervalo. Por fim, você deve garantir ainda que a todo momento as pontuações são **distintas** (não há empates).

Input

A primeira linha contém dois inteiros N e Q ($1 \leq N \leq 1000$, $0 \leq Q \leq 10^5$) — o número de tuk-tuks e o número de atualizações observadas.

As próximas Q linhas contêm dois inteiros P_i e X_i , indicando que o tuk-tuk P_i passou a ocupar a posição X_i . É garantido que $1 \leq P_i, X_i \leq N$ e que X_i é uma posição melhor ou igual a posição que P_i estava antes da manobra ser registrada.

Output

A primeira linha deve conter N inteiros: a pontuação inicial de cada tuk-tuk (do 1 ao N), separados por espaço.

Em seguida, imprima Q linhas, cada uma com um inteiro representando o número de pontos ganhos pelo respectivo tuk-tuk naquela manobra.

Todos os valores impressos devem estar entre 0 e 10^9 (inclusive).

Examples

input
3 2 2 1 3 2
output
2 1 0 3 3

input
5 5 5 1 4 1 3 1 2 1 1 1
output
4 3 2 1 0 5 5 5 5 5



Durante a maratona de programação em Campo Grande, em 2022, Jake Peralta e seu amigo Charles Boyle aproveitaram para visitar o Bioparque Pantanal, um dos maiores aquários de água doce do mundo. Lá, ficaram encantados com a variedade de peixes das mais diversas espécies.

Enquanto observavam os tanques, Jake teve uma ideia para um jogo: Boyle escolheria aleatoriamente um peixe (com probabilidade igual entre todos), e Jake teria que adivinhar qual era, fazendo apenas perguntas com resposta "sim" ou "não".

Jake poderia perguntar o que quisesse: "O nome do peixe tem um número par de letras?", "O peixe tem escamas azuis?", "O peixe é nativo do Brasil?", e por aí vai...

Jake, sendo extremamente estratégico, decidiu seguir um plano ótimo, minimizando o número esperado de perguntas necessárias para descobrir o peixe escolhido. Mas, como ele não é tão bom assim com contas, pediu sua ajuda. Sua tarefa é: dado a lista dos nomes dos peixes do aquário, todos com nomes distintos, calcular o valor esperado de perguntas que Jake precisará fazer se usar a estratégia ótima.

Input

A primeira linha contém um número inteiro N ($1 \leq N \leq 10^5$), o número de peixes.

As próximas N linhas contêm os nomes distintos dos peixes. Cada nome é composto apenas por letras do alfabeto latino (maiúsculas e/ou minúsculas). A soma total dos comprimentos dos nomes não excede 10^6 .

Output

Imprima o número esperado de perguntas necessárias. O erro absoluto ou relativo permitido é de no máximo 10^{-4} .

Examples

input
10 salmao atum sardinha tilapia bacalhau panga merluza

dourado dori nemo
output
4.4000000000

input
1 peixe
output
1.0000000000

input
2 atum tilapia
output
2.0000000000

Note

Após reduzir as opções para apenas um peixe, Jake ainda precisa de um turno para confirmar a sua resposta. Por exemplo, mesmo quando $N = 1$ ele deve fazer uma pergunta. Veja os exemplos para mais detalhes.

I. Moscou

time limit per test: 1 second
memory limit per test: 512 megabytes



Bonecas Matriosca. Ilustrativo. Na questão as bonecas podem estar fora de ordem e estão em uma única linha.

Durante sua visita a Moscou, na Rússia, em 2021, durante a Final Mundial da ICPC, Arthur entrou em uma charmosa loja de antiguidades e se deparou com uma vitrine repleta de *bonecas matrioscas*, famosas bonecas russas que podem ser colocadas umas dentro das outras. As bonecas estavam dispostas em linha, mas não seguiam nenhuma ordem de tamanho. Algumas eram maiores, outras menores, e algumas tinham exatamente o mesmo tamanho.

Arthur decidiu comprar algumas bonecas, escolhendo um conjunto contínuo das bonecas para levar para casa. Porém, como Arthur terá que despachar a mala, ele começou a se preocupar com a *resistência* do grupo de bonecas ao

transporte. Arthur definiu que a **força** de um grupo de bonecas é calculada da seguinte forma. Seja T_i o tamanho da i -ésima boneca, a força do conjunto de bonecas começando da boneca l até a boneca r é dado por:

$$F = (r - l + 1)^2 \cdot \min_{k=l}^r \{T_k\}$$

Ou seja, a quantidade de bonecas ao quadrado vezes o tamanho da menor boneca.

Não é a toa que Arthur está na final mundial, então calcular qual seria o grupo mais forte ou mais fraco é fácil demais para Arthur. Ao invés disso, Arthur quer escolher o grupo com a K -ésima menor força.

Formalmente, os tamanhos de cada boneca são dados como um array de N inteiros T_i . Dentre todos as $\frac{N(N+1)}{2}$ formas de escolher uma sequência contínua de bonecas (i.e. subarrays), qual é aquela com a K -ésima menor força do grupo?

Input

A primeira linha contém dois inteiros N e K ($1 \leq N \leq 10^5, 1 \leq K \leq \frac{N(N+1)}{2}$) — o número de bonecas na vitrine e o valor K desejado.

A segunda linha contém N inteiros $T_1, T_2, \dots, T_N$ ($1 \leq T_i \leq 10^7$) — os tamanhos das bonecas da esquerda para a direita.

Output

Imprima um único inteiro: a força do grupo que está na K -ésima posição entre todos os grupos possíveis, quando ordenados em ordem crescente de força.

Note que esse número pode ser muito grande.

Examples

input
4 5 2 1 3 2
output
4

input
3 4 1 1 1
output
4

input
3 6 1 1 1
output
9



A Final brasileira da maratona de programação de 2021 foi sediada na charmosa cidade de Gramado, no Rio Grande do Sul. Ao final da prova, ficou claro pelo placar que a competição havia sido particularmente difícil: o time campeão resolveu 8 problemas, e o segundo lugar — ainda com medalha de ouro — resolveu 6, mesmo número de problemas resolvido até a 16ª colocação.

Com tantos desafios, frustrações e clima nublado, muitos participantes decidiram se consolar da melhor forma possível que Gramado pode oferecer: **com chocolate quente**.

Entre eles estava Naim, um dos juízes responsáveis por criar as questões da prova. Já de barriga cheia de chocolates, ele começou a divagar sobre a vida, sobre a prova... e então teve um pensamento completamente aleatório:

"E se eu tivesse que escolher K lembranças daquela prova — representadas por pares de valores A_i (quanto aprendi) e B_i (quanto doeu) — de forma que a soma total dos aprendizados menos o maior sofrimento fosse a mais alta possível?"

Por mais louco que pareça, Naim agora precisa da sua ajuda para resolver esse problema de forma urgente, antes que ele tome mais um chocolate quente!

Formalmente, lhe são dados N pares de inteiros (A_i, B_i) . Um conjunto S tem valor de aprendizado dado por $v = \sum_{i \in S} A_i - \max_{i \in S} B_i$. Para todo K de 1 até N você deve achar o valor máximo de aprendizado que um conjunto S de tamanho K pode ter.

Input

A primeira linha contém um inteiro N ($1 \leq N \leq 2 \times 10^5$).

Cada uma das próximas N linhas contém dois inteiros A_i e B_i ($-10^9 \leq A_i \leq 10^9$, $-2 \times 10^{14} \leq B_i \leq 2 \times 10^{14}$).

Output

Imprima N linhas. A K -ésima linha deve conter o valor máximo de aprendizado considerando todos os subconjuntos de tamanho K .

Examples

input
3 6 1 7 6 5 2
output
5 9 12

input
2 1000 1 1000 1
output
999 1999

input
6 7 7 0 4 5 1 -1002 0 1 4 6 5
output
4 6 11 12 12 -990

K. Online

time limit per test: 1 second
memory limit per test: 256 megabytes



Durante a pandemia de 2020, a Final Brasileira da Maratona de Programação teve que ser realizada de forma online. Antes do início da competição, os participantes podiam acessar o site, mas a senha só seria liberada quando a competição começasse. Curioso com segurança e programação, você decide testar um ataque clássico de tempo conhecido como *timing attack*.

A ideia desse tipo de ataque é que, frequentemente, ao comparar duas strings, a função de verificação retorna falso assim que encontra um caractere incorreto no prefixo. Isso significa que quanto mais caracteres corretos no início da string você fornecer, maior tende a ser o tempo de resposta. Este problema é inspirado nesse princípio.

```
// Funcao que compara se duas strings sao iguais, parando quando algum valor no prefixo for diferente.
int isEqual(const std::string& a, const std::string& b) {
    if(a.size() != b.size())return 0;
    for (int i = 0; i < a.size(); ++i) {
        if (a[i] != b[i]) {
            return 0;
        }
    }
    return 1;
}
```

Você precisa descobrir uma senha binária oculta, representada por uma string S de tamanho $1 \leq N \leq 1000$, usando apenas consultas interativas.

Input

Leia apenas um inteiro N , o tamanho da string.

Interaction

- A cada consulta, você envia uma string binária de tamanho N .
- O juiz irá comparar sua string com a senha S e retornará um inteiro T' com base na quantidade real de caracteres corretos T do maior **prefixo em comum**.
- O valor retornado T' será:
 - $T - 1, T$ ou $T + 1$ com alguma probabilidade fixa (não necessariamente uniforme).
 - Se $T = 0$, o retorno será sempre 1 (nunca 0 ou negativo).
 - Se $T = 1$, o retorno será sempre 1 ou 2 (nunca 0).
 - Se a senha estiver correta, o juiz retornará 1000000 e você pode encerrar o seu programa.
 - Se você exceder o número de consultas permitidas ou fizer uma consulta incorreta o juiz retornará -1 .

Você pode utilizar no máximo 3500 consultas.

O algoritmo para obter o maior prefixo em comum é dado abaixo:

```
// a.size() == N e b.size() == N
int commonPrefix(const std::string& a, const std::string& b){
    int correct = 0;
    while(correct < N && a[correct] == b[correct])
        correct++;
    return correct;
}
```

Examples

input
3
1
3
1000000
output
000
100

101

input

4

2

1000000

output

1001

1000

Note

Essa questão possui 248 testes, todos com $N = 1000$, com exceção dos samples.

No primeiro exemplo:

- A senha é 101.
- Primeiro lemos 3 da entrada.
- Em seguida, é perguntando se a senha é 000. O maior prefixo é $T = 0$, mas é impresso o valor de 1.
- Depois é perguntado 100 com $T = 2$. É impresso $T + 1 = 3$.
- Por fim acertamos a senha: 101.

No segundo exemplo:

- A senha é 1000.
- Primeiro lemos 4 da entrada.
- Em seguida, é perguntando se a senha é 1001. O maior prefixo é $T = 3$, mas é impresso o valor de $T - 1 = 2$.
- Por fim acertamos a senha: 1000.

Note que os exemplos são ilustrativos, e não foram gerados por uma das soluções oficiais.

L. Campina Grande

time limit per test: 3.5 seconds

memory limit per test: 1024 megabytes



A final brasileira de 2019 foi realizada em Campina Grande, Paraíba, onde estudou uma lenda viva da maratona brasileira. Na UFCG, estudou o lendário FMota, conhecido por muitos como o Fastest Coder Alive Mota após ter conseguido codar o Dijkstra de ponta a ponta em menos de 30 segundos. Desde muito jovem, ele participa de competições de programação e acumula diversos prêmios como: duas medalhas de ouro na OBI Sênior, campeão da primeira final Latino americana no México e até Campeão da competição Café com Leite (CCL) 2024 em João Pessoa.

Mas com tanta fama, vêm os problemas, e um deles é o guarda-roupa cheio demais. Isso porque, a cada competição que participa, o FMota ganha uma camiseta exclusiva. Como ele compete todos os anos, ele já tem uma enorme coleção, que cresce cada ano mais e mais.

Sendo fissurado por números, o FMota gosta de calcular quantas formas distintas ele pode vestir todas as suas camisetas durante os próximos dias até esgotar o guarda roupa. Ele faz isso respeitando uma regra pessoal:

FMota utiliza uma camiseta por dia, e nunca repete camisetas até que tenha utilizado todas do guarda roupa. Além disso, ele veste as camisetas de cada competição em blocos, ou seja, primeiro usa todas da OBI (sem repetir, em qualquer ordem), depois todas da Maratona (sem repetir), e assim por diante.

Dado que ele possui um número inicial de camisetas de cada competição, representado pelo vetor C , FMota quer saber quantas formas diferentes ele pode vestir suas camisetas nos próximos $K + 1$ anos (o ano atual mais K anos futuros). A cada novo ano, ele ganha exatamente uma nova camiseta de cada competição. Assim, no ano 0 ele tem C_i camisetas da competição i , no ano 1 ele terá $C_i + 1$, no ano 2 terá $C_i + 2$, e assim por diante.

Sua tarefa é calcular quantas formas FMota pode vestir as camisas no ano 0 até o ano K . Como esse número é muito grande, imprima a resposta módulo 998244353. Ou seja, se existem x formas, você deve imprimir um inteiro $0 \leq x' < 998244353$ representando o resto da divisão inteira de x por 998244353.

Input

A primeira linha contém dois inteiros N e K ($1 \leq N \leq 10^5$, $0 \leq K \leq 10^5$) — o número de competições que FMota participa regularmente e a quantidade de anos no futuro a considerar.

A segunda linha contém N inteiros $C_1, C_2, \dots, C_N$ ($0 \leq C_i \leq 10^5$) — o número atual de camisetas que FMota tem de cada competição.

Output

Imprima uma linha com $K + 1$ inteiros, separados por espaço, representando o número de formas que FMota pode se vestir módulo 998244353.

Examples

input
1 5 0
output
1 1 2 6 24 120

input
3 4 5 3 7
output
3628800 696729600 854310693 788947910 701240539

input
2 2 3 3
output
36 576 14400

