Problema 1 (Chave Numérica)

Breno e Bruno são especialistas em buscas! Em uma misteriosa sala existem diversos cofres, com tesouros inestimáveis em uma delas. No entanto, a chave de cada cofre é mantida em segredo, e apenas com uma estratégia inteligente é possível encontrá-la.

Breno e Bruno encontraram um pedaço de papel ao lado de um dos cofres com a seguinte pista:

10 20
> < > = (7)

Eles então inseriram o número 7 no cofre próximo ao local em que encontraram o pedaço de papel e o cofre abriu. Porém, para a infelicidade deles, não havia tesouros no cofre. Eles então perceberam que ao lado dos cofres haviam pedaços de papéis, porém sem o resultado para abrir o cofre – número entre parênteses.

A partir da pista, eles descobriram que a chave numérica é um número inteiro positivo e se encontra no intervalo limitado entre 10 e 20, inclusive. Eles também descobriram que a sequência de símbolos maior (>), menor (<) ou igual (=) é um **conjunto mínimo de operações** a serem realizadas para encontrar o valor 7 no intervalo.

Cabe a você agora criar um programa que, dado o intervalo e o conjunto de operações a serem realizadas, descubra qual é a chave de cada cofre na sala, ajudando Breno e Bruno.

Entrada
A entrada consistirá em um número inteiro N positivo e não mais que 10 indicando a quantidade de casos de teste.

Para cada caso de teste, duas linhas contendo os seguintes números e símbolos:
A B
“Conjunto de símbolos”

Onde A e B são respectivamente os limites inferior e superior do intervalo, e o conjunto formado pelos símbolos ‘>’, <’ ou ‘=’, indicando o mínimo de respostas. Para todos os efeitos 1<= A < B <= 1000.

Saída
A saída deve exibir a chave numérica que abre o cofre de cada intervalo, com um salto de linha após cada chave.

Exemplo de entrada
2
1 100
< < < > < =
10 20
> < > =

Exemplo de saída
7
17

Problema 2 (massa)

Um composto orgânico é qualquer membro de uma grande classe de compostos químicos cujas moléculas contêm carbono. A *massa molar* de um composto orgânico é a massa de um mol do composto orgânico. A massa molar de um composto orgânico pode ser calculada a partir dos pesos atômicos padrão dos elementos.

Dr. CHON deseja encontrar sua massa molar de compostos orgânicos a partir de *fórmulas moleculares*. A fórmula molecular mostra quais são os elementos que formam determinada substância e o número exato de átomos de cada elemento que está presente em uma molécula dessa substância.. Por exemplo, na *fórmula molecular* $C_3H_4O_3$ temos 3 átomos de carbono (C), 4 átomos de Hidrogênio (H) e 3 átomos de oxigênio (O).

Neste problema, assumimos que a fórmula molecular é representada por apenas quatro elementos, ‘C’ (Carbono), 'H' (Hidrogênio), 'O' (Oxigênio) e 'N' (Nitrogênio) sem parênteses. A tabela a seguir mostra os pesos atômicos padrão para 'C', 'H', 'O' e 'N'.

Nome Atômico	Carbono	Hidrogênio	Oxigênio	Nitrogênio
Peso atômico padrão	12.01 g/mol	1.008 g/mol	16.00 g/mol	14.01 g/mol

Por exemplo, a massa molar de uma fórmula molecular C_6H_5OH é 94,108 g/mol que é calculada por 6 × (12,01 g/mol) + 6 × (1,008 g/mol) + 1 × (16,00 g/mol). Dada uma *fórmula molecular*, escreva um programa para calcular a massa molar da fórmula.

Entrada:
Seu programa deve ler da entrada padrão. A entrada consiste em T casos de teste. O número de casos de teste T é dado na primeira linha da entrada. Cada caso de teste é dado em uma única linha, que contém uma fórmula molecular como uma *string*. O símbolo químico é dado por uma letra maiúscula e o comprimento da *string* é maior que 0 e menor que 80. O número de quantidade n que é representado após o símbolo químico seria omitido quando o número for 1 ($2 \leq n \leq 99$) .

Saída:
Seu programa deve escrever na saída padrão. Imprima exatamente uma linha para cada caso de teste. A linha deve conter a massa molar da fórmula molecular dada.

Exemplo de Entrada:
4
C
C6H5OH
NH2CH2COOH
C12H22O11

Exemplo de saída:
12.010
94.108
75.070
342.296

Problema 3. (Jogo de Cartas)

Em um emocionante jogo de cartas, os jogadores recebem cartas numeradas e o objetivo é obter a maior pontuação possível seguindo algumas regras. Cada carta possui um número inteiro.

As regras para calcular a pontuação são as seguintes:

- O jogador escolhe uma sequência de cartas consecutivas (pode ser uma sequência de apenas uma carta).
- A pontuação da sequência é a soma dos números das cartas na sequência.
- No entanto, o jogador não pode incluir duas cartas consecutivas em sua sequência se ambas tiverem o mesmo valor.

Por exemplo, suponha uma configuração com 16 cartas conforme abaixo:

13 -3 -25 20 -3 -16 -23 **18 20 -7 12** -5 -22 15 -4 7

A sequência que de cartas consecutivas que dá o maior valor está em negrito (**18 20 -7 12**) que resulta em 43

Sua tarefa é desenvolver um programa que determine a maior pontuação que um jogador pode obter dada uma sequência de cartas.

Entrada:
A entrada consiste de vários casos de teste. Cada caso de teste consiste em uma configuração de n cartas, cada uma com o seguinte formato:
n
a_1 a_2 a_3 ... a_n

Onde:

- n é o número de cartas na configuração, 0 < n <= 100;
- a_i é o número na i-ésima carta, –500 <= a_i <= 500;

A entrada termina quando uma configuração com 0 cartas aparecer.

Saída:
A saída deve exibir a maior pontuação possível para a configuração de cartas, com um salto de linha no final.

Exemplo de entrada:
16
13 -3 -25 20 -3 -16 -23 18 20 -7 12 -5 -22 15 -4 7
12
-420 250 150 -320 480 -190 -470 360 80 -230 -280 420
0

Exemplo de Saída:
43
560

Problema 4. (Ruínas de SJ)

Nas antigas ruas de paralelepípedos da cidade histórica de São João del Rei, exploradores desvendam segredos esquecidos e subterrâneos. As ruínas possuem mecanismos únicos, com dispositivos que podem ser considerados armadilhas para os exploradores. Você, um especialista em enigmas históricos, deve criar um programa para analisar dispositivos, verificar se estão configurados corretamente e se são seguros para explorar.

Cada dispositivo tem símbolos de ativação e desativação. Ativações usam símbolos como '(' para pistas, '[' para bloqueios e '{' para passagens. Desativações usam símbolos correspondentes: ')' para pistas, ']' para bloqueios e '}' para passagens.

Para um dispositivo ser considerado seguro, a ordem de emparelhamento deve ser respeitada, ou seja, uma ativação só pode ser fechada por uma desativação correspondente.

O desafio é criar um algoritmo que verifique a configuração dos dispositivos. Se a configuração estiver correta, o programa exibe "Seguro!" Caso contrário, a saída é "Armadilha!".

Entrada:
Seu programa deve ler da entrada padrão. A entrada consiste em um conjunto de dispositivos. A primeira linha do arquivo de entrada contém o número de dispositivos a serem analisados, que é um número inteiro positivo e não maior que 20. As linhas subsequentes descrevem as configurações de cada dispositivo.

Saída
Para cada dispositivo, exiba "Seguro!" ou "Armadilha!", com uma quebra de linha ao final de cada resposta.

Exemplo de entrada
3
{[()]}
{[(])}
{{[[(())]]}}

Exemplo de Saída
Seguro!
Armadilha!
Seguro!

Problema 5 (labirinto)

Sua missão, se você decidir aceitá-la, é criar um programa de desenho de labirinto. Um labirinto consistirá nos caracteres alfabéticos A-Z, * (asterisco) e espaços.

Entrada

Seu programa obterá as informações dos labirintos do arquivo de entrada. Este arquivo conterá linhas de caracteres que seu programa deve interpretar para desenhar um labirinto. Cada linha do labirinto será descrita por uma série de números e caracteres, onde os números antes de um caractere informam quantas vezes esse caractere será usado. Se houver vários dígitos em um número antes de um caractere, o número de vezes para repetir o caractere será a soma dos dígitos antes desse caractere. A letra minúscula 'b' será usada no arquivo de entrada para representar espaços no labirinto. As descrições de diferentes linhas no labirinto serão separadas por um ponto de exclamação (!), ou por um final de linha. Descrições para diferentes labirintos serão separadas por uma linha em branco. O arquivo de entrada será encerrado por um fim de arquivo.

Saída

Para cada descrição no arquivo de entrada, desenhe o labirinto correspondente conforme mostrado no exemplo de saída abaixo. Não há limite para o número de linhas em um labirinto ou para o número de labirintos em um arquivo, embora nenhuma linha contenha mais de 132 caracteres. Imprima uma linha em branco entre dois labirintos consecutivos.

Feliz labirinto!

Exemplo de Entrada
1T1b5T!1T2b1T1b2T!1T1b1T2b2T!1T3b1T1b1T!3T3b1T!1T3b1T1b1T!5T1*1T

11X21b1X
4X1b1X

Exemplo de Saída

```
T TTTTT
T  T TT
T T  TT
T   T T
TTT   T
T   T T
TTTTT*T

XX   X
XXXX X
```

Problema 6 (gene)

Uma maneira que os cientistas têm para tentar medir como uma espécie evoluiu para outra é investigando como o genoma do ancestral se modificou para se transformar nesta outra espécie. Espécies intimamente relacionadas têm vários genes em comum e verifica-se que uma boa maneira de compará-las é através da comparação de como os genes comuns mudaram de lugar.

Uma das mutações mais comuns que alteram a ordem dos genes de genomas é a inversão. Se modelarmos um genoma como uma sequência de N genes sendo cada gene um número inteiro de 1 a N,então uma inversão é uma mutação que altera o genoma revertendo a ordem de um bloco de genes consecutivos. A inversão pode ser descrita por dois índices (i, j), (1 ≤ i ≤ j ≤ N), indicando que ela inverte a ordem dos genes dentro de índices de i até j.

Assim, quando isto é aplicado para um genoma [g1,. . . , gi-1, gi , gi+1,. . . , gj-1, gj , gj+1,. . . , gN], obtém-se o genoma [g1,. . . , gi-1, gj , gj-1,. . . , gi+1, gi , gj+1,. . . , gN]. Como um exemplo, a inversão de (3, 6), aplicado à genoma [1, 2, 3, 4, 5, 6, 7] dá [1, 2, 6, 5, 4, 3, 7]. Se depois que a inversão (1, 3) é aplicada, obtém-se o genoma [6, 2, 1, 5, 4, 3, 7].

Um cientista que está estudando a evolução de uma espécie deseja tentar uma série de inversões no genoma desta espécie. Em seguida, ele quer consultar a posição final de vários genes. Será que você aceita o desafio de ajudá-lo?

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N indicando o número de genes no genoma (1 ≤ N ≤ 50000). Você pode supor que o ordem inicial dos genes é a sequência de números inteiros de 1 a N em ordem crescente. A segunda linha de um caso de teste contém um inteiro R (0 ≤ R ≤ 1000) que indica o número de inversões a serem aplicadas ao genoma. Então, R linhas seguem, cada uma contendo dois inteiros i, j (1 ≤ i ≤ j ≤ N), separados por um único espaço, o qual indicam os dois índices que definem a inversão correspondente. Após a descrição das inversões há uma linha contendo um inteiro Q (0 ≤ Q ≤ 100), que indica o número de consultas para os genes, seguido de Q linhas, onde cada linha contém um inteiro representando um gene cuja posição final você deve determinar.

O final da entrada é indicada por N = 0.

Saída

Para cada caso de teste da entrada seu programa deve produzir Q + 1 linhas de saída. A primeira linha deve conter a string "Genome", seguido do número do caso de teste. As seguintes Q linhas devem conter um número inteiro, cada um representando as respostas das consultas.

Exemplo de Entrada

```
9
1
3 6
4
1
3
5
1
5
2
1 2
1 5
2
5
2
0
```

Exemplo de Saída

```
Genome 1
1
6
4
1
Genome 2
1
5
```

Problema 7 (Mistério nas Estrelas)

No universo hipotético de Primarium, a galáxia é repleta de sistemas estelares, planetas exóticos e fenômenos cósmicos. Os astrônomos de Primarium observaram um fenômeno intrigante: sempre que um cometa passa pelo sistema Alpha Prismática, ele emite sinais codificados que parecem estar relacionados aos fatores primos do número de dias que levam para completar sua órbita ao redor da estrela.

Os cientistas de Primarium estão ansiosos para entender essa conexão e desvendar os segredos que os cometas estão transmitindo através desses sinais. Para isso, eles precisam de um programa que ajude a determinar os fatores primos dos números de dias que os cometas levam para completar suas órbitas.

Dado um número inteiro positivo D, representando o número de dias que um cometa leva para completar sua órbita, você deve escrever um programa para determinar todos os fatores primos desse número.

Entrada
A entrada consiste de vários números inteiros positivos D (2 <= D <= 550), representando o número de dias que cada cometa leva para completar sua órbita. Toda entrada finaliza com um número 0.

Saída
Para cada número da entrada, imprima todos os fatores primos de D, separados por um espaço. Ao final de cada conjunto de fatores, existe um salto de linha.

Exemplo de entrada:
180
42
90
0

Exemplo de saída:
2 2 3 3 5
2 3 7
2 3 3 5