278 - Xadrez – 94%

Quase todo mundo conhece o problema de colocar oito rainhas em um tabuleiro de xadrez 8 x 8 de tal forma que nenhuma rainha ameace outra. Jan Timman (um famoso enxadrista holandês) gostaria de saber qual é a maior quantidade de peças de xadrez de um tipo que pode ser colocada em um tabuleiro de m x n casas de forma que uma peça não ameace a outra. Já que é difícil fazer esses cálculos manualmente, ele pediu a sua ajuda para resolver este problema.

Ele não precisa saber a resposta para todas as peças. Peões são desinteressantes e ele também não gosta de Bispos. Ele deseja saber apenas quantas Torres (*Rooks*), Cavalos (*Knights*), Rainhas (*Queens*) e Reis (*Kings*) podem ser colocados no tabuleiro de forma que uma peça não ameace a outra.

Entrada

A primeira linha da entrada contém o número de problemas. Um problema é será colocado em cada uma das linhas seguintes e consiste de um caractere para representar o tipo de peça: `r`, `k`, `Q`, `K`, significando, respectivamente, Rook, Knight, Queen ou King. O caractere será seguido por dois inteiros: m (4 <= m <= 10) e n (4 <= n <= 10) representando o número de linhas e colunas do tabuleiro.

Saída

Para cada problema de entrada seu programa deverá imprimir o número máximo de peças de xadrez que podem ser colocadas no tabuleiro indicado de forma que uma peça não ameace a outra.

Nota: Um tabuleiro 1x1 é um tabuleiro com apenas uma casa. Não é necessário construir nenhum algoritmo de tentativa e erro ou equivalente, basta encontrar uma “fórmula” para cada tipo de peça.

Exemplo de Entrada

```
2
r 6 7
k 8 8
```

Exemplo de Saída

```
6
32
```

575 – Binário Distorcido – 97%

Quando um número é expresso na forma decimal, o k-ésimo dígito representa um múltiplo de 10^k. (Dígitos são numerados da direita para a esquerda, onde o dígito menos significativo é o número 0.), por exemplo:

$$\mathbf{81307_{10} = 8x10^4 + 1x10^3 + 3x10^2 + 0x10^1 + 7x10^0 = 80000 + 1000 + 300 + 0 + 7 = 81307}$$

Quando um número é expresso na forma binária, o k-ésimo digito representa um múltiplo de 2^k. Por exemplo:

$$\mathbf{10011_2 = 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 16 + 0 + 0 + 2 + 1 = 19}$$

Em **Binário Distorcido**, o k-ésimo dígito representa um múltiplo de $2^{k+1} - 1$. Os únicos dígitos possíveis são 0 e 1, exceto o dígito menos-significativo diferente de zero, o qual pode ser 2. Por exemplo:

$$\mathbf{10120_{skew} = 1x(2^5 - 1) + 0x(2^4 - 1) + 1x(2^3 - 1) + 2x(2^2 - 1) + 0x(2^1 - 1) = 31 + 0 + 7 + 6 + 0 = 44}$$

Os 10 primeiros números em Binário Distorcido são 0, 1, 2, 10, 11, 12, 20, 100, 101, e 102. (Binário Distorcido é útil em algumas aplicações porque é possível adicionar 1 com no máximo um "vai um". No entanto, isto não tem nada a ver com o problema atual.)

Entrada

A entrada conterá uma ou mais linhas, cada uma contendo um inteiro n. Se n = 0 isto significará o fim da entrada, caso contrário n será um inteiro não negativo no formato Binário Distorcido.

Saída

Para cada número da entrada, imprima a forma decimal equivalente. O valor decimal será no máximo $2^{31} - 1 = 2147483647$.

Exemplo de Entrada

```
10120
2000000000000000000000000000000
10
1000000000000000000000000000000
11
100
1111100000111000010110110200
0
```

Exemplo de Saída

```
44
2147483646
3
2147483647
4
7
1041110737
```

1203 - Argus - 96%

Um stream de dados é uma sequência contínua, em tempo real e ordenada de itens. Alguns exemplos incluem sensores de dados, tráfego na internet, atividades financeiras, leilões online, logs de transações realizadas via Web e registros de ligações telefônicas. Da mesma forma, consultas sobre streams executam continuamente sobre um período de tempo e retornam novos resultados incrementalmente na medida em que novos dados são recebidos. Por exemplo, um sistema de detecção de temperatura de uma fábrica pode ter as seguintes consultas executando:

Consulta-1: A cada 5 minutos, recupere a temperatura máximo dos últimos 5 minutos.

Consulta-2: Retorne a temperatura média em cada andar nos últimos 10 minutos.

Nós desenvolvemos um Sistema Gerenciador de Stream de Dados chamado Argus, que processa consultas sobre stream de dados. Usuários podem registrar as consultas no Argus e o sistema manterá as consultas executando sobre os dados e retornará os resultados correspondentes na frequência desejada.

Para o Argus, nós usamos a seguinte instrução para registrar uma consulta: ***Register Q_num Period***

sendo: *Q_num* ($0 < Q\_num \leq 3000$) o identificador da consulta e *Period* ($0 < Period \leq 3000$) corresponde ao intervalo entre dois retornos consecutivos de resultados. Em *Period* segundos após o registro da consulta, o resultado será retornado pela primeira vez e, depois disso, o resultado será retornado a cada *Period* segundos.

Existirão diversas consultas registradas no Argus de cada vez. Todas as consultas terão um identificador *Q_num* diferente. Sua tarefa é informar quais serão as primeiras K consultas que retornarão seus resultados. Se duas ou mais consultas irão retornar seus resultados ao mesmo tempo, os resultados são retornados um a um e ordenados de maneira crescente em relação ao *Q_num*.

Entrada

A primeira parte da entrada contém o registro das instruções para o Argus, uma instrução por linha. Você pode assumir que o número de instruções não será maior que 1000 e todas essas instruções são lidas ao mesmo tempo. Esta parte da entrada é encerrada com uma linha contendo apenas "#".

A segunda parte contém uma linha com o valor de K ($K \leq 10000$). Correspondendo ao número de linhas de saída que deverão ser impressas.

Saída

Você deverá imprimir o *Q_num* das primeiras *K* consultas a retornarem seus resultados, um número por linha.

Exemplo de Entrada

```
Register 2004 200
Register 2005 300
#
5
```

Exemplo de Saída

```
2004
2005
2004
2004
2005
```

1225 – Soma de Dígitos – 94%

Trung está entediado com suas lições de casa de matemática. Ele pega um pedaço de giz e começa a escrever sequências de inteiros consecutivos iniciando do 1 até N (1 < N < 10000). Depois disso, ele conta o número de vezes que cada dígito (de 0 a 9) apareceu na sequência. Por exemplo, com N=13 temos a seguinte sequência:

12345678910111213

Nesta sequência, 0 aparece uma vez, 1 aparece seis vezes, 2 aparece duas vezes, 3 aparece duas vezes e cada dígito do 4 ao 9 aparecem apenas uma vez. Depois de um tempo, Trung ficou entediado novamente. Ele agora quer escrever um programa para fazer isso para ele. Sua tarefa é ajudá-lo a escrever este programa.

Entrada

A entrada corresponde a vários casos de teste. A primeira linha da entrada contém o número de casos de teste e corresponde a um inteiro positivo menor ou igual a 20. As linhas seguintes descrevem os casos de teste. Para cada caso de teste há uma única linha contendo o número N.

Saída

Para cada caso de tese, escreva uma linha contendo, sequencialmente, o número de vezes que cada dígito ocorre do zero até o nove, separados por um espaço em branco.

Exemplo de Entrada

2
3
13

Exemplo de Saída

0 1 1 1 0 0 0 0 0 0
1 6 2 2 1 1 1 1 1 1

1230 – MODEX – 96%

Muitos operadores criptográficos bem conhecidos necessitam da exponenciação modular. Isto é, dados os inteiros, x, y e n, compute $\mathbf{x^y \bmod n}$. Neste problema, você deverá implementar de maneira eficiente esta operação.

Entrada

A entrada consiste de um linha contendo o número de casos de teste c, seguido por c conjuntos de teste, seguidos por uma linha contendo o número "0".

Cada conjunto de teste consiste de uma única linha contendo três inteiros x, y e n, separados por espaços em branco. Você pode assumir que: $1 < x , n < 2^{15} = 32768$, e $0 < y < 2^{31} = 2147483648$.

Saída

A saída consiste de uma linha para cada caso de teste. A iésima linha deverá conter z, tal que:

$$z = x^y \bmod n$$

para os respectivos números x , y , z dados no iésimo caso de teste.

Exemplo de Entrada

```
2
2 3 5
2 2147483647 13
0
```

Exemplo de Saída

```
3
11
```

1260 – Vendas – 99%

O senhor Cooper, o CEO da CozyWalk Co., recebe um relatório sobre as vendas diárias da companhia desde que ela foi criada. A partir do segundo dia da criação da empresa, ao receber o relatório, ele compara o relatório atual com cada um dos relatórios anteriores a fim de calcular o número de dias nos quais os valores são menores ou iguais aos atuais. Depois de obter esse número de dias, ele escreve o valor em uma lista.

Este problema pode ser descrito mais formalmente da seguinte maneira. Sendo $A = (a_1, a_2,..., a_n)$ e denotando a lista de quantidades vendidas diariamente; e sendo $B = (b_1, b_2,..., b_{n-1})$ a lista de inteiros mantida pelo senhor Cooper, cada valor representando o número de dias anteriores que obedecem a seguinte regra. No i-ésimo dia ($2<=i<=n$), b_{i-1} é calculado da seguinte forma: o número de a_k's tal que $a_{k<=}a_i$ ($1<=k < i$).

Por exemplo, sendo $A = (20, 43, 57, 43, 20)$. Para o quarto dia temos que a quantidade de vendas $a_4 = 43$; e o número de dias anteriores nos quais a quantidade de vendas foi menor ou igual a este valor é 2, pois $a_{1<=}a_4$, $a_{2<=}a_4$, e $a_3 > a_4$. Assim, $b_3 = 2$. Similarmente, b_1, b_2, e b_4 podem ser calculados, obtendo-se: $B = (1, 2, 2, 1)$.

Dado um arranjo de tamanho n com a lista de vendas diárias, escreva um programa que imprima a soma dos n-1 inteiros presentes na lista B.

Entrada

Seu programa deverá ler a entrada que consistirá de T casos de teste. O número de casos de teste T é dado na primeira linha da entrada. Cada caso de teste será iniciado com uma linha contendo o inteiro n ($2<=n<=1000$), que representa o tamanho da lista A. Na linha seguinte haverá n inteiros, cada um representando a quantidade diária de vendas a_i ($1<=a<=5000$ e $1<=i<=n$) para o caso de deste.

Saída

Seu programa deverá imprimir, para cada caso de teste, a soma dos n-1 elementos presentes na lista B (obtida a partir da lista A). A seguir há um exemplo de entrada e de saída que contém dois casos de teste.

Exemplo de Entrada

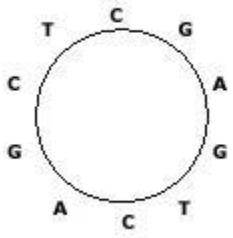
```
2
5
38 111 102 111 177
8
276 284 103 439 452 276 452 398
```

Exemplo de Saída

```
9
20
```

1584 - Sequência Circular – 96%

Algumas sequências de DNA são circulares, como mostrado pela figura a seguir que contém a sequência circular "CGAGTCAGCT", o último caractere "T" em "CGAGTCAGCT" é conectado ao primeiro caractere "C". Nós sempre lemos sequências circulares no sentido horário.



Já que não é fácil armazenar uma sequência circular em um computador, nós decidimos armazená-la como uma sequência linear. No entanto, pode haver muitas sequências lineares que são obtidas a partir de uma sequência circular (caso você inicie a sequência linear de diferentes pontos da sequência circular). Assim, nós também decidimos armazenar apenas a sequência linear que é lexicograficamente a menor entre todas as possíveis (dada uma sequência circular).

Sua tarefa é encontrar a menor sequência linear (lexicograficamente falando) a partir de uma dada sequência circular. Por exemplo, a menor sequência linear a partir da sequência circular da figura é "AGCTCGAGTC".

Entrada

A entrada consistirá de T casos de teste. O número T é dado na primeira linha da entrada. Cada caso de teste terá apenas uma linha contendo a sequência circular que estará representada por uma de suas possíveis representações lineares. Já que nossas sequências são de DNA, haverá apenas quatro símbolos (caracteres) possíveis: A, C, G e T. Cada sequência terá de 2 a 100 caracteres.

Saída

Imprima uma linha para cada caso de teste contendo a sequência linear lexicograficamente menor da respectiva sequência circular. A seguir são apresentados exemplos de entrada e saída (contendo dois casos de teste).

Exemplo de Entrada

```
2
CGAGTCAGCT
CTCC
```

Exemplo de Saída

```
AGCTCGAGTC
CCCT
```