Maratoninha

2ª Competição – 27 Agosto 2016

início : 13:30 Término : 17:30

Organização e Realização

Antonio Marcos M Hachisuca(coordenação geral)
Alcione Banacchio, Estevan Braz Brandt Costa, Charles Juca Busarello (coordenadores IFPR)
Equipe Organizadora :
Claudio Roberto M. Mauricio, Eliane Pereira Nascimento, Alan Lino dos Reis, Christian F. dos Santos, Flávio H. Canesso, Gustavo dos Santos Vieira, Isshak M. Darwich, Leonardo F. Scalco, Paulo Moggi,William A. da Rosa.

Lembretes:

- Aos javaneiros: **o nome da classe deve ser o mesmo nome do arquivo a ser submetido.** Ex: **classe petrus**, nome do arquivo **petrus.java;**

- Exemplo de leitura de entradas que funcionam:
  Java: (import java.util.Scanner)
  Scanner in = new Scanner(System.in);
  ou
  Scanner stdin = new Scanner(new BufferedReader(new InputStreamReader(System.in)));
  C: (#include <stdio.h>)
  int integer1; scanf("%d", &integer1);
  C++: (#include <iostream>)
  int integer1; std::cin >> integer1;
  Pascal : Readln(a,b);

  Exemplo de saída de entradas:

  Java: System.out.format("%d %d\n", integer1, integer2);
  C: printf("%d %d\n", integer1, integer2);
  C++: std::cout << integer1 << " " << integer2 << std::endl;
  Pascal: writeln(a,b);

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova;
- A correção é automatizada, portanto, **siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos.** Deve-se considerar entradas e saídas padrão;
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido . . .), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos;
- Utilize o clarification para dúvidas da prova. Os juízes podem **opcionalmente** atendê-lo com respostas acessíveis a todos;

Patrocinadores e Agradecimentos

- EITS – Patrocinador oficial
- AIS – Patrocinador oficial
- Centro Maker – Patrocinador oficial
- Fundação PTI-BR – Patrocinador oficial
- ITAIPU Binacional – Patrocinador oficial
- CACC – Centro Acadêmico Compuatção – apoio
- IFPR Foz – Parceiro da competição
- Aos voluntários pelo empenho e dedicação
- Ao grupo GNU Linux pelo apoio ao projeto

Problema A

Cadeias de Palíndromos

Arquivo fonte: palin.c, palin.cpp ou palin.java

Uma cadeia de caracteres é chamada de palíndromo se sequência de caracteres da esquerda para a direita é igual à sequência de caracteres da direita para a esquerda (uma outra definição é que o primeiro caractere da cadeia deve ser igual ao último caractere, o segundo caractere seja igual ao penúltimo caractere, o terceiro caractere seja igual ao antepenúltimo caractere, e assim por diante). Por exemplo, as cadeias de caracteres 'mim', 'axxa' e 'ananaganana' são exemplos de palíndromos.
Se uma cadeia não é palíndromo, ela pode ser dividida em cadeias menores que são palíndromos.
Por exemplo, a cadeia 'aaxyx' pode ser dividida de quatro maneiras distintas, todas elas contendo apenas cadeias palíndromos: {'aa', 'xyx'}, {'aa', 'x', 'y', 'x'}, {'a', 'a', 'xyx'} e {'a', 'a', 'x', 'y', 'x'}.

1. Tarefa

Escreva um programa que determine qual o menor número de partes em que uma cadeia deve ser dividida de forma que todas as partes sejam palíndromos.

2. Entrada

A entrada é constituída de vários conjuntos de teste. A primeira linha de um conjunto de testes contém um inteiro N que indica o número de caracteres da cadeia ($1 \leq N \leq 2000$). A segunda linha contém a cadeia de caracteres, composta por letras minúsculas (de 'a' a 'z'), sem espaços em branco. O final da entrada é indicado por N = 0.

Exemplo de Entrada

```
3
axa
6
xyzyyx
10
bbabcbbaab
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste n", onde n é numerado a partir de 1. A segunda linha deve conter um inteiro indicando o menor número de partes que a cadeia de entrada deve ser dividida de forma que todas as partes sejam palíndromos. A terceira linha deve ser deixada em branco. O formato mostrado no exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
1

Teste 2
4

Teste 3
4
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 2000$ (N = 0 apenas para indicar o fim da entrada)

Problema B

Parque Jurássico

Arquivo fonte: parque.c, parque.cpp ou parque.java

O DNA é uma molécula envolvida na transmissão de caracteres hereditários e na produção de proteínas, que são os principais constituintes de seres vivos. O DNA é formado pelas bases nitrogenadas adenina (A), guanina (G), citosina (C) e timina (T).
A identificação da sequência de bases que constitui uma determinada parte do DNA pode ajudar a descoberta da cura de doenças que atacam seres vivos. Teoricamente, a identificação do DNA pode também permitir a recriação de espécies extintas, como na estória do escritor americano Michael Crichton.
O professor de Biologia de sua escola, prof. Estevão Espilbergo, conseguiu amostras de células de uma espécie de mosquito extinto a milhares de anos, e pretende, ambiciosamente, recriar o animal a partir de seu DNA. Para isso, conseguiu que um laboratório de genômica fizesse a identificação das bases das células. No entanto, pelo estado precário das células obtidas, o resultado não foi dos melhores. O professor Estevão recebeu do laboratório duas sequências, com a informação de que essas sequências contêm, provavelmente, muitos “buracos”, ou seja, entre uma base e outra corretamente detectadas podem existir bases não detectadas.
O prof. Estevão então decidiu combinar as duas sequências para formar uma sequência única, e precisa de sua ajuda.

1. Tarefa

Sua tarefa é escrever um programa que determine a menor sequência que contenha, como subsequências, as duas sequências obtidas pelo laboratório. Dizemos que uma sequência S1 é subsequência de uma outra sequência S2 se acrescentando-se alguns elementos a S1 obtém-se S2. Por exemplo, ACGT é uma subsequência de ATCGAAT, pois basta inserir um T após o A e dois A’s após o G.

2. Entrada

A entrada possui vários conjuntos de teste. Cada conjunto de teste é composto por duas linhas, cada uma contendo uma sequência S composta por caracteres ‘A’, ‘C’, ‘G’ e ‘T’. O final da entrada é indicado por uma linha contendo o caractere ‘#’.

Exemplo de Entrada

```
AAATTT
GAATCT
ACGT
ATCGAAT
#
```

3. Saída

Para cada conjunto de teste, o seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste n”, onde n é numerado sequencialmente a partir de 1. A segunda linha deve conter uma sequência de comprimento mínimo que contenha as duas sequências da entrada como subsequências. Se houver mais de uma sequência de comprimento mínimo, seu programa pode escrever qualquer uma delas. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

Teste 1
GAAATCTT

Teste 2
ATCGAAT

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

1 ≤ número de caracteres de S ≤ 100

Problema C

Jogo do Bicho

Arquivo fonte: `bicho.c`, `bicho.cpp`, `bicho.java` *ou* `bicho.pas`

Em um país muito distante, as pessoas são viciadas em um jogo de apostas bastante simples. O jogo é baseado em números e é chamado *jogo do bicho*. O nome do jogo deriva do fato que os números são divididos em 25 grupos, dependendo do valor dos dois últimos dígitos (dezenas e unidades), e cada grupo recebe o nome de um animal. Cada grupo é associado a um animal da seguinte forma: o primeiro grupo (burro) consiste nos números 01, 02, 03 e 04; o segundo grupo (águia) é composto dos números 05, 06, 07 e 08; e assim em diante, até o último grupo contendo os números 97, 98, 99 e 00.

As regras do jogo são simples. No momento da aposta, o jogador decide o valor da aposta V e um número N ($0 \leq N \leq 1000000$). Todos os dias, na praça principal da cidade, um número M é sorteado ($0 \leq M \leq 1000000$). O prêmio de cada apostador é calculado da seguinte forma:

- se M e N têm os mesmos quatro últimos dígitos (milhar, centena, dezena e unidade), o apostador recebe $V \times 3000$ (por exemplo, $N = 99301$ e $M = 19301$);
- se M e N têm os mesmos três últimos dígitos (centena, dezena e unidade), o apostador recebe $V \times 500$ (por exemplo, $N = 38944$ e $M = 83944$);
- se M e N têm os mesmos dois últimos dígitos (dezena e unidades), o apostador recebe $V \times 50$ (por exemplo, $N = 111$ e $M = 552211$);
- se M e N têm os dois últimos dígitos no mesmo grupo, correspondendo ao mesmo animal, o apostador recebe $V \times 16$ (por exemplo, $N = 82197$ and $M = 337600$);
- se nenhum dos casos acima ocorrer, o apostador não recebe nada.

Obviamente, o prêmio dado a cada apostador é o máximo possível de acordo com as regras acima. No entanto, não é possível acumular prêmios, de forma que apenas um dos critérios acima deve ser aplicado no cálculo do prêmio. Se um número N ou M com menos de quatro dígitos for apostado ou sorteado, assuma que dígitos 0 devem ser adicionados na frente do número para que se torne de quatro dígitos; por exemplo, 17 corresponde a 0017.

Dado o valor apostado, o número escolhido pelo apostador, e o número sorteado, seu programa deve calcular qual o prêmio que o apostador deve receber.

Entrada

A entrada contém vários casos de teste. Cada caso consiste em apenas uma linha, contendo um número real V e dois inteiros N e M, representando respectivamente o valor da aposta com duas casas decimais ($0.01 \leq V \leq 1000.00$), o número escolhido para a aposta ($0 \leq N \leq 1000000$) e o número sorteado ($0 \leq M \leq 1000000$). O final da entrada é indicado por uma linha contendo $V = M = N = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada um dos casos de teste seu programa deve imprimir uma linha contendo um número real, com duas casas decimais, representando o valor do prêmio correspondente a aposta dada.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
32.20 32 213929 10.50 32 213032 2000.00 340000 0 520.00 874675 928567 10.00 1111 578311 0 0 0	515.20 5250.00 6000000.00 0.00 500.00

Problema D

Abelha

Arquivo fonte: abelha.c, abelha.cpp ou abelha.java

Na África há uma espécie muito especial de abelhas. Todos os anos, as abelhas fêmeas dessas espécies dão à luz a uma abelha macho, enquanto que as abelhas macho dão a luz a uma abelha macho e uma fêmea, e então eles morrem!

Agora os cientistas descobriram acidentalmente uma "abelha fêmea mágica" de tais espécies especiais sendo que ela é imortal, mas ainda capaz de dar à luz uma vez por ano, como todas as outras abelhas fêmeas. Os cientistas gostariam de saber quantas abelhas existirão depois de N anos. Escreva um programa para ajudá-los a encontrar o número de abelhas macho e o número total de todas as abelhas, após N anos.

Entrada
Cada linha da entrada contém um inteiro N ($N \geq 0$). A entrada finaliza quando N = -1 (neste caso, não deve ser processado).

Saída
Cada linha da saída deverá ter dois números, o primeiro é o número de abelhas macho após N anos, e o segundo representa o total de abelhas após N anos. (Os dois números não deverão exceder 2^{32}).

Exemplo de entrada

```
1
3
-1
```

Exemplo de saída

```
1 2
4 7
```

Problema E

Armstrong

arquivo: armstrong.c , armstrong.cpp, armstrong.java

Dado um intervalo de números inteiros positivos, você deve encontrar aqueles números que podem ser escritos como a soma de cada um de seus dígitos individuais elevados ao número de seus dígitos.

Um número *N é conhecido como um número Armstrong de ordem n (n sendo o número de dígitos)* se
$abed..... = a^n + b^n + e^n + d^n + ... = N$

Exemplo:
$370 = 3^3 + 7^3 + 0^3 = 27 + 343 + 0 = 370$

$9474 = 9^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 9474$

Entrada
A entrada contém vários casos de testes. A primeira linha da entrada contem um número T ($1 \leq T \leq 1000$) que representa o número de casos de testes. As próximas T linhas contêm dois números separados por um espaço, A1 e A2, onde A1 ≤ A2 ($0 \leq A1 \leq A2 \leq 2^{31}$) , correspondendo ao inicio e fim do intervalo fechado onde a busca será efetuada.

Saída
Para cada caso de teste, deverá ser gerada uma única linha. A linha conterá os números de Armstrong encontrados separados por um espaço (considere um espaço após o último elemento no final da linha), ou caso não seja encontrado nenhum número, deverá ser escrita a palavra "Nenhum".

Exemplo de Entrada
```
2
1 9
100 110
```

Exemplo de Saída
```
1 2 3 4 5 6 7 8 9
Nenhum
```

Problema F
Margaridas

Nome do arquivo fonte: margaridas.c, margaridas.cpp, ou margaridas.java

Leopoldo é gerente de uma plantação de flores da Associação de Cultivo de Margaridas (ACM), um grupo que cultiva margaridas em grandes propriedades para abastecer floriculturas em grandes cidades.
As margaridas são plantadas em vasos dispostos em linhas e colunas, formando uma espécie de grade. Na plantação administrada por Leopoldo existem L linhas de vasos de margaridas, cada uma formada por C vasos.
Para facilitar o gerenciamento, os vasos são organizados em lotes de M linhas e N colunas de vasos, sendo que não existem sobreposições entre os lotes (não existe nenhuma linha ou coluna comum a mais de um lote) e todos os lotes têm exatamente M linhas e N colunas.
A colheita é sempre feita em um único lote, coletando-se todas as margaridas daquele lote que estejam prontas para a venda. Uma semana antes de fazer a colheita, os funcionários da plantação analisaram cada vaso e anotaram quantas margaridas estarão prontas para venda na semana seguinte. Leopoldo agora precisa da sua ajuda para determinar qual o número máximo de margaridas que poderá ser colhido em um único lote de M × N vasos.

Tarefa

Sua tarefa é escrever um programa que, dado um mapa da plantação contendo o número de margaridas prontas para venda em cada vaso, encontre qual o número máximo de margaridas que podem ser colhidos por Leopoldo.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (normalmente o teclado). A primeira linha da entrada contém quatro números inteiros, L, C, M e N. L e C representam respectivamente o número de linhas (1 ≤ L ≤ 1000) e de colunas (1 ≤ C ≤ 1000) de vasos existentes na plantação. M e N representam respectivamente o número de linhas (1 ≤ M ≤ L) e de colunas (1 ≤ N ≤ C) dos lotes. As L linhas seguintes contêm C inteiros cada, representando número de margaridas prontas para colheita no vaso localizado naquela linha e coluna.

Saída

Seu programa deve imprimir, na saída padrão, uma única linha que contém o número máximo de margaridas que podem ser colhidos em um lote de M × N. Esse número não pode ser superior a 1000000.

Entrada	Entrada	Entrada
3 3 1 1 1 2 3 1 3 3 1 10 1 Saída 10	4 4 2 1 1 2 3 4 5 6 7 8 1 10 5 2 1 5 9 10 Saída 16	5 5 2 2 1 1 1 3 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 3 1 1 3 Saída 7

Problema G

Criptografia por Bases

arquivo: base.c , base.cpp, base.java

Natal é um cientista da computação apaixonado por números. Recentemente lhe ocorreu uma idéia inovadora: é possível aplicar a teoria de conversão de bases na área de criptografia.

Todas as bases podem ser representadas por um alfabeto de símbolos em ordem crescente, na qual o primeiro símbolo equivale ao zero. Por exemplo: o sistema decimal pode ser representado por "0123456789", o binário por "012", a hexadecimal por "0123456789ABCDEF" e assim por diante.

Dessa maneira, poderíamos utilizar qualquer alfabeto desde que a representação obedeça as regras citadas acima. Uma estranha base poderia ser composta pelo alfabeto "oF8", onde o = 0 e o < F < 8, dessa maneira os números de um a dez da base decimal seriam convertidos para F, 8, Fo, FF, F8, 8o, 8F, 88, Foo, FoF.

O processo de criptografia de uma mensagem necessita de duas bases, a de criptografia (utilizada na mensagem criptografada) e a de apresentação (utilizada para reconstrução da mensagem original).

Você foi convidado a participar desse ambicioso projeto, que pode revolucionar a computação contemporânea. Para isso será necessário desenvolver o algoritmo de "descriptografia". Dada uma mensagem, a base de criptografia e a base de apresentação, o algoritmo deverá ser capaz de descobrir a mensagem original.

Entrada

A primeira linha de código é um inteiro N (1 ≤ N ≤ 100) que indica o número de casos de teste. Cada uma das N linhas posteriores terá 3 valores separados por espaço: ***mensagem base_criptografia base_apresentação***, onde 1 ≤ número de símbolos na mensagem ≤ 40, 2 ≤ número de símbolos em base_criptografia ≤ 100 e 2 ≤ número de símbolos em base_apresentação ≤ 100. Cada base é definida com um alfabeto de símbolos dispostos em ordem crescente. Não há símbolos repetidos na representação de nenhuma das bases. Os símbolos podem ser: números (0-9), letras (a-zA-Z) ou um dos caracteres a seguir `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Saída

Para cada caso de teste deverá ser impressa uma linha de código contendo "`Caso #x: `", onde x é o número do caso de teste, seguido da mensagem descriptografada.

Exemplo de Entrada

```
3
9 0123456789 oF8
Foo oF8 0123456789
13 0123456789abcdef 01
```

Exemplo de Saída

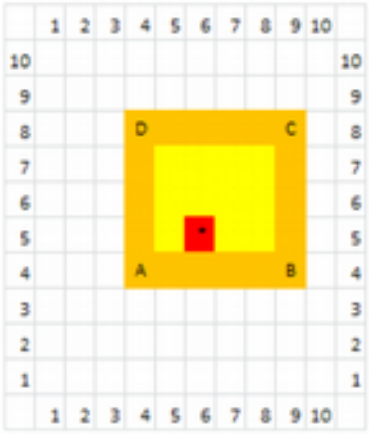
```
Caso #1: Foo
Caso #2: 9
Caso #3: 10011
```

Problema H

Colisão

arquivo: colisao.c , colisao.cpp, colisao.java

Você recebeu a missão de verificar se o robô invadiu uma área retangular formada por quatro pontos cardeais (A,B,C e D). Serão informados os quatro pontos de um plano cardeal conforme a figura. A área será formada pela ligação dos quatro pontos da seguinte forma A-B, B-C, C-D e D-A. Será informado ainda a coordenada X,Y do robô.



Entrada

A entrada é composta de vários casos de testes. A primeira linha é formada por um número N indicando o total de casos de testes. As próximas N linhas são constituídas por 10 números inteiros (A_x, A_y, B_x, B_y, C_x, C_y, D_x, D_y, R_X, R_Y) representando cada um dos vértices A, B, C e D e pela posição X, Y do robô. Cada valor é separado por um espaço em branco.

Saída

A saída deverá imprimir para cada caso de testes o número 1, se o robô estiver dentro da área, e imprimir o número 0 caso contrário.

Exemplo de Entrada

```
5
3 6 6 6 6 5 3 5 5 4
1 1 7 1 7 7 1 7 4 2
1 4 7 4 7 6 1 6 5 5
6 2 9 2 9 6 6 6 1 7
4 3 9 3 9 5 4 5 10 7
```

Exemplo de Saída

```
0
1
1
0
0
```

Problema I

O Problema da Parada

Nome do arquivo fonte: `parada.c`, `parada.cpp`, `parada.java` *ou* `parada.pas`

O Problema da Parada (*The Halting Problem*) é um problema de decisão clássico da Ciência da Computação que consiste, basicamente, em determinar se um dado programa sempre vai parar (ou seja, terminar sua execução) para uma dada entrada arbitrária ou se vai executar infinitivamente. Alan Turing provou, em 1936, que é impossível resolver o problema da parada generalizando para qualquer par programa-entrada. Neste problema, porém, dada a descrição de uma linguagem simples, um programa escrito nessa linguagem e uma entrada para esse programa, você deve determinar se o programa dado pára com a entrada dada e, em caso positivo, qual a saída produzida.

Esta linguagem só trabalha com números inteiros de `0` a `999` (inclusive). Sendo assim, o sucessor de `999` é `0`, e o antecessor de `0` é `999`. Além disso, ela possui dez variáveis (`R0` a `R9`), sendo que a `R0` sempre é atribuído o valor de chamada do programa (ou seja, o parâmetro de entrada) e a `R9` é sempre atribuído o valor de saída (o retorno). No início da execução do programa, é atribuído o valor `0` a todas as variáveis, com exceção de `R0` que recebe o parâmetro de entrada.

As operações básicas são atribuição (`MOV`), soma (`ADD`), subtração (`SUB`), multiplicação (`MUL`), divisão inteira (`DIV`) e resto da divisão inteira (`MOD`). Todas essas operações têm a sintaxe `COMANDO OPERANDO1,OPERANDO2` (sem espaços entre a vírgula e os operandos), onde `COMANDO` é uma dessas operações, `OPERANDO1` é uma das 10 variáveis (`R0` a `R9`) e `OPERANDO2` pode ser uma das 10 variáveis ou um valor inteiro (entre `0` e `999`). Todas as operações modificam o valor de `OPERANDO1`, sendo assim `MOV R4,100` é o equivalente a atribuir o valor `100` a `R4`, enquanto que `MUL R3,R8` é o equivalente a multiplicar `R3` por `R8` e atribuir o resultado a `R3`. A operação `DIV`, assim como a `MOD`, retornam `0` (zero) se `OPERANDO2` for `0` ou se a variável equivalente tiver valor `0`. Ou seja, `DIV R4,0` é o equivalente a `MOV R4,0`. Por divisão inteira, entendemos a parte inteira do quociente da divisão (sem a parte fracionária). Por exemplo, a divisão inteira de `7` por `2` é `3` (sendo o resto `1`).

Existem seis comandos de fluxo de decisão: `IFEQ` (se igual), `IFNEQ` (se diferente), `IFG` (se maior), `IFL` (se menor), `IFGE` (se maior ou igual) e `IFLE` (se menor ou igual). A sintaxe de todos eles é `COMANDO OPERANDO1,OPERANDO2` (sem espaços entre a vírgula e os operandos), onde `OPERANDO1` e `OPERANDO2` podem ser variáveis (`R0` a `R9`) ou valores inteiros (entre `0` e `999`). Assim, o comando `IFEQ R4,123` é o equivalente a testar se `R4` é igual a `123`. Caso a condição testada seja verdadeira, o programa continua a executar normalmente a linha subseqüente ao comando de decisão. Caso a condição seja falsa, o programa passa a executar a linha subseqüente ao `ENDIF` mais próximo. **Todos** os comandos de decisão devem ter um comando `ENDIF` correspondente.

Finalmente, existem os comandos `CALL` e `RET`, ambos com a sintaxe `COMANDO OPERANDO`, onde `OPERANDO` é uma variável (`R0`..`R9`) ou valor direto (entre `0` e `999`). O comando `CALL` chama o próprio programa novamente, passando `OPERANDO` como parâmetro de entrada, ou seja, atribuindo o valor de OPERANDO à variavel R0. Já `RET` termina a execução do programa, retornando o valor de `OPERANDO` como o resultado de saída. **A última linha do programa sempre será um comando** `RET`. Observe que, caso o programa chame a si mesmo através do comando `CALL`, quando a execução voltar, o valor de `R9` vai estar alterado com o valor retornado pelo programa. Note também que **todas** as variáveis (`R0`..`R9`) são *locais*, ou seja, uma chamada subseqüente ao programa não pode alterar os valores guardados nas variáveis da

instância anterior, com exceção, naturalmente, do valor de R9 que recebe o retorno da instância chamada.

O exemplo a seguir ilustra um programa que calcula o fatorial de um número.

linha	comando
1	IFEQ R0,0
2	RET 1
3	ENDIF
4	MOV R1,R0
5	SUB R1,1
6	CALL R1
7	MOV R2,R9
8	MUL R2,R0
9	RET R2

1a linha: Verifica se o valor de R0 vale 0, caso positivo, executa a próxima linha, caso contrário, pula para a 4a linha (ENDIF mais próximo).

2a linha: Retorna 1 como saída do programa.

3a linha: Marca o fim do bloco de decisão iniciado na primeira linha.

4a linha: Atribui o valor de R0 a R1 (R1 ← R0).

5a linha: Diminui 1 de R1 (R1 ← R1 - 1).

6a linha: Chama o programa passando R1 como parâmetro de entrada.

7a linha: Guarda o valor de R9 (retornado pela chamada anterior) em R2 (R2 ← R9)

8a linha: Multiplica o valor de R2 por R0 (R2 ← R2 * R0)

9a linha: Retorna o valor de R2 como saída do programa.

A tabela seguir traz um resumo dos comandos para referência:

comando	sintaxe	significado
MOV	MOV OP1,OP2	OP1 ← OP2
ADD	ADD OP1,OP2	OP1 ← OP1 + OP2
SUB	SUB OP1,OP2	OP1 ← OP1 - OP2
MUL	MUL OP1,OP2	OP1 ← OP1 * OP2
DIV	DIV OP1,OP2	OP1 ← OP1 / OP2
MOD	MOD OP1,OP2	OP1 ← OP1 % OP2
IFEQ	IFEQ OP1,OP2	if OP1 == OP2
IFNEQ	IFNEQ OP1,OP2	if OP1 != OP2
IFG	IFG OP1,OP2	if OP1 > OP2
IFL	IFL OP1,OP2	if OP1 < OP2
IFGE	IFGE OP1,OP2	if OP1 >= OP2
IFLE	IFLE OP1,OP2	if OP1 <= OP2
ENDIF	ENDIF	Marca fim do bloco de execução condicional
CALL	CALL OP	Chama o programa com OP como entrada
RET	RET OP	return OP

Entrada

A entrada contém vários casos de teste. Cada caso de teste se inicia com dois inteiros, L e N, representando respectivamente o número de linhas do programa ($1 \leq L < 100$) e o valor do parâmetro de entrada do programa ($0 \leq N < 1000$). As L linhas seguintes contêm o programa. Pode-se assumir que ele está sempre sintaticamente correto de acordo com as regras definidas acima. Todos os comandos (bem como o nome das variáveis) só conterão letras maiúsculas. O final da entrada é marcado pelo caso em que $L = N = 0$ e não deve ser processado.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste, seu programa deve produzir uma linha contendo um inteiro que representa o valor de saída (retorno) para a entrada N dada, ou um asterisco (*) no caso de o programa nunca terminar.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**

9 6
IFEQ R0,0
RET 1
ENDIF
MOV R1,R0
SUB R1,1
CALL R1
MOV R2,R9
MUL R2,R0
RET R2
2 123
CALL R0
RET R0
0 0
	720
	*

Problema J

E agora Mamá ?

Arquivo fonte: mama.c, mama.cpp ou mama.java

Na velha China, o mestre ThaiTchunCamon, era conhecido pelos súditos apenas como Camon. Camon tinha um conselheiro muito inteligente e admirado por toda a região, seu nome era Mamá.
Camon convocou Mama para resolver uma questão: preciso saber de quantas maneiras diferentes eu consigo subir as escadarias da muralha da China, só que, indo de 1 em 1 de degraus, ou de 2 em 2. Para ajudar, mestre Camon disse brilhantemente que, para subir até o primeiro degrau tem somente uma possibilidade, e para chegar ao segundo degrau, tem duas possibilidades.
Você deve ajudar Mamá a resolver esta questão.

Entrada
A entrada é composta por vários casos de testes.
A primeira linha é composta de um inteiro N ($N \leq 10000$) que representa o número de casos de testes.
Cada caso de teste é composto por um número inteiro C ($3 \leq C \leq 10000$) indicando o número de degraus a serem escalados.

Saída
Em cada linha deverá ser impresso o número de possibilidades de subir a escadaria.

Exemplo de entrada

```
2
3
5
```

Exemplo de saída

```
3
8
```

Problema K

Quadro Premiado

arquivo: quadro.c , quadro.cpp, quadro.java

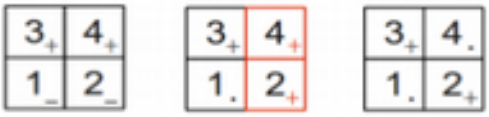
Você está em um programa de televisão, e tem uma ótima chance de ganhar muito dinheiro. Trata-se de um jogo com algumas regras peculiares, e o montante de dinheiro resultante dependerá apenas da sua esperteza, podendo-se até sair perdendo caso se jogue mal.

O jogo funciona da seguinte maneira: há um quadro, com N linhas e M colunas, e em cada posição deste quadro há um inteiro positivo, representando uma quantia em dinheiro. Em cada uma dessas posições você tem a opção de colocar um dos seguintes sinais:

'+' - Significa que o valor daquela posição deve ser somado à seu prêmio.
'-' - Significa que o valor daquela posição deve ser subtraído do seu prêmio.
'.' - Significa que tal posição deve ser ignorada.

A vida seria muito simples se você pudesse colocar '+' em todas as posições, portanto há duas regras adicionais ao jogo: para cada linha do quadro, você deve preencher as posições com um dos padrões de sinais montados pelos organizadores do jogo; e para cada coluna do quadro, não é permitido que duas posições adjacentes verticalmente tenham o mesmo sinal (se aplica aos sinais '+' e '-'). É possível usar o mesmo padrão mais de uma vez, desde que não desrespeitando a segunda regra acima.

Veja um exemplo na imagem abaixo, onde os padrões são: "++", "--", ". +" e "+ .".



Considere que há sempre ao menos uma maneira de se completar o quadro.

Como o jogo é novo, eles deixaram que você usasse seu computador para te ajudar na decisão, sem saber que você era um programador. Escreva um algoritmo que lhe diga qual a soma máxima que é possível alcançar no jogo.

Entrada

Haverá diversos casos de teste. Cada caso de teste inicia com dois inteiros, N e M ($1 \leq N, M \leq 100$), indicando o número de linhas e de colunas do quadro, respectivamente.
A seguir haverá N linhas, contendo M inteiros cada, representando os valores do quadro. Seja v o valor de qualquer posição do quadro, $1 \leq v \leq 100$.
A seguir haverá um inteiro K ($1 \leq K \leq 100$), indicando o número de padrões. Em seguida haverá K linhas, cada uma com M caracteres, representando cada um dos padrões, conforme a simbologia descrita no enunciado.
O último caso de teste é indicado quando N = M = 0, o qual não deverá ser processado.

Saída

Para cada caso de teste imprima uma linha, contendo um inteiro, representando a soma máxima que é possível alcançar se os padrões forem escolhidos de forma ótima.

Exemplo de Entrada

```
2 2
3 4
1 2
4
++
--
```

```
+.
.+
3 3
1 3 2
4 2 3
3 5 1
2
+.+
-+-
0 0
```

Exemplo de Saída

```
5
8
```