OBI 2015

(SIMULADO COLÉGIO ARI DE SÁ CAVALCANTE)

CADERNO DE TAREFAS

Modalidade **Programação** - Níveis **1** e **2**, Fase **1**

25 de abril de 2015

A PROVA TEM DURAÇÃO DE 5 HORAS

Promoção:

Patrocínio:

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 29 páginas (não contando com a folha de rosto) numeradas de 1 a 21. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao help do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída do seu programa.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python devem ser arquivos com sufixo *.py*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
- Não utilize arquivos para cada entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln*, *read*, *writeln*, *write*;
  - em C: *scanf*, *getchar*, *printf*, *putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo, Scanner, BufferedReader, BufferedWriter e System.out.println
  - em Python: read, *readline*, *readlines*, *print*, *write*
- Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

1. TORRES DE HANÓI

Nome do arquivo fonte: hanoi.c, hanoi.cpp, hanoi.pas, hanoi.java, hanoi.py

A Torre de Hanói é um "quebra-cabeça" que foi divulgado pela primeira vez pelo matemático francês Édouard Lucas que consiste em uma base contendo três pinos, onde em um deles estão dispostos alguns discos uns sobre os outros em ordem crescente de diâmetro de cima para baixo. O problema consiste em passar todos os discos de um pino para outro qualquer, usando um dos pinos como auxiliar, de maneira que o disco maior nunca fique em cima de outro menor em nenhuma situação.

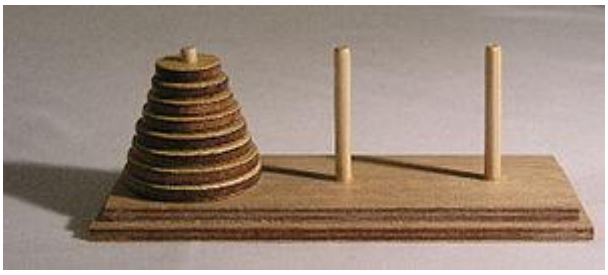

Figura 1. Exemplo de uma Torre de Hanói.

1.1 Seja n o número de discos, escreva um programa que determine o número mínimo de movimentos necessários para conseguir transferir todos os discos da primeira estaca para a terceira. Resolva o problema para vários casos de teste. O programa é encerrado caso o número de discos seja zero.

Entrada
Número de discos. O programa é finalizado com a entrada 0.

Saída
A primeira linha deve conter o número de teste, no formato “Teste n”, sem as aspas. A linha seguinte deve conter o número mínimo de movimentos para o respectivo caso.

Exemplo
Entrada
3
1
0

Saída
Teste 1
7
Teste 2
1

1.2 Escreva uma solução recursiva para o problema das torres de hanoi.

2. SEQUÊNCIA DE FIBONACCI

Nome do arquivo fonte: fibonacci.c, fibonacci.cpp, fibonacci.pas, fibonacci.java, fibonacci.py

A Sequência (ou Sucessão) de Fibonacci é uma sequência numérica de inteiros que possui como valores inicias 1, 1 e que é definida como a soma dos dois números imediatamente anteriores, ou seja:

$$F_n = F_{n-1} + F_{n-2}$$

Onde F_1=1, F_2=1, F_3=2, F_4= 3 e assim sucessivamente, ou seja:

1, 1, 2, 3, 5, 8, 13, ...

Ela também pode ser definida pela explicitamente pela Fórmula de Binet:

$$F(n) = \frac{1}{\sqrt{5}}\left\{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right\}$$

Fazendo n tender a infinito n na razão $\frac{F(n+1)}{F(n)}$, encontraremos um valor chamado proporção áurea, cujo valor é dado por:

$$\lim_{n\to\infty} \frac{F(n+1)}{F(n)} = \emptyset = \frac{1+\sqrt{5}}{2} = 1{,}6180$$

A sequência de Fibonacci e a Razão Áurea são facilmente encontradas na natureza, como no na espiral do Nautilus, anatomia do corpo humano, arranjos de folhas além de obras de arte e arquitetônicas:

Figura 2. Da esquerda para direita: Nautilus, Panteão em Athenas, Grécia e Mona Lisa de Leonardo da Vinci.

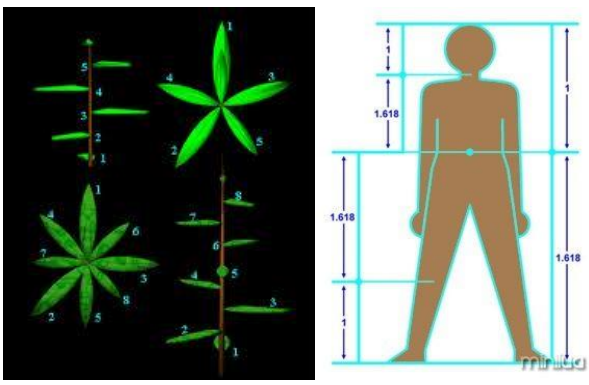

Figura 3. Sequência de Fibonacci em Arranjo de Folhas e Proporção Áurea no Corpo Humano.

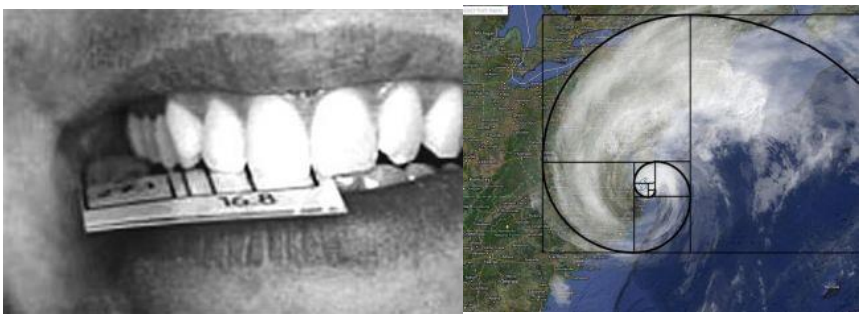

Figura 4. Razão Áurea na Arcada Dentária e Sequência de Fibonacci em Furacões.

Sua tarefa é escrever um programa que, digitado determinado o número dos n primeiros elementos desejados da sequência, seja impresso na tela o primeiro até o n-ésimo elemento. O programa é encerrado com entrada zero.

Entrada
Número de elementos da sequência de Fibonacci desejado.

Saída
A primeira linha deve conter o número de teste, no formato "Teste n", sem as aspas. A linha seguinte deve conter o os n primeiros elementos da sequência de Fibonacci.

Exemplo
Entrada
6
10
0

Saída
Teste 1
1 1 2 3 5 8
Teste 2
1 1 2 3 5 8 13 21 34 55

3. INTEGRAL DE RIEMANN

Nome do arquivo fonte: integral.c, integral.cpp, integral.pas, integral.java, integral.py

Denomina-se integral de Riemann de uma função *f(x)* no intervalo *[a, b]* a área sob a curva definida por *f(x)* no intervalo *[a, b]* (ver Figura 5).

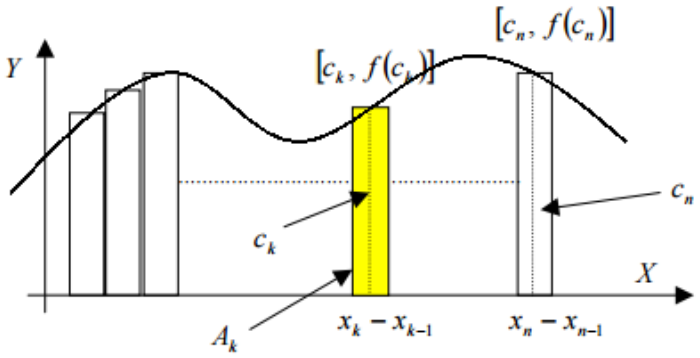

Figura 5. Área sob a curva *f(x)*.

O que matematicamente, para o caso discreto, escrevemos por:

$$\int_a^b f(x)dx = \lim_{\Delta x_k \to 0} \sum_{k=1}^{n} f(c_k)\,\Delta x_k, \qquad \Delta x_k = x_k - x_{k-1}$$

Portanto, o cálculo numérico da integral de Riemman é realizado da seguinte forma: pega-se *n* retângulos com base fixa tão pequena quanto possível (Δx_k), de tal forma que, seja c_k o início do intervalo, temos então que o primeiro retângulo terá área $\Delta x_1 c_1$, o segundo retângulo terá área $\Delta x_2 c_2$, a terceira área $\Delta x_3 c_3$ e assim por diante.

Tarefa

Sua tarefa é calcular a integral de Riemman de uma função do segundo grau $f(x)=ax^2+bx+c$ em um intervalo $I=[x_1,x_2]$ fornecidos pelo usuário.

Entradas

Coeficientes da equação do segundo grau;
Limites do intervalo *I*;

Observação

O valor da base do retângulo deverá ser menor possível. Utilize um valor que julgar adequado (ou seja, será arbitrado pelo programador). Vale ressaltar que quanto menor o valor de Δx_k, maior será a precisão da resposta, entretanto, mais demorado será o algoritmo.

Tarefa-Extra:

Refaça o algoritmo para a curva $f(x)= x+1$ para o intervalo fechado [0,1]. Sabendo que o valor da integral de *f(x)* para o dado intervalo é 1,5, execute o algoritmo e calcule o erro absoluto para os seguintes casos:

(a) $\varepsilon=0,5$
(b) $\varepsilon=0,1$
(c) $\varepsilon=0,01$
(d) $\varepsilon=0,001$

4. IRA

Nome do arquivo fonte: ira.c, ira.cpp, ira.pas, ira.java, ira.py

Na Universidade Federal do Ceará (UFC), o rendimento do aluno é medido pelo Índice de Rendimento Acadêmico (IRA). O IRA é calculado da seguinte forma:

$$IRA = \frac{\sum_{i=1}^{n} Cr_i \cdot N_i}{\sum_{i=1}^{n} Cr_i}$$

Onde, Cr_i define o crédito da i-ésima disciplina, N_i define a nota da i-ésima disciplina, ou seja, o IRA é a média ponderada das notas nas *n* disciplinas cujos pesos são os créditos.

Tarefa

Sabendo que o desvio-padrão do IRA de uma turma é dada por:

$$\sigma = \sqrt{\frac{1}{a-1}\sum_{i=1}^{a}(IRA_i - \overline{IRA})^2}$$

Onde IRA_i é o IRA do i-ésimo aluno e que $\overline{IRA}$ é o IRA médio de todos os *a* alunos de uma referida turma. Pede-se: Faça um algoritmo que, para *x* turmas, cada uma contendo um determinado número *x* de alunos que cursaram um determinado número *n* de disciplinas com determinado número de créditos *Cr* e nota *N*:

- Calcule o IRA individual de todos os alunos;
- Calcule o IRA médio da turma e o desvio-padrão;
- Exiba o maior e o menor IRA.

Entrada

- Número de turmas;
- Número de estudantes e disciplinas cursadas em cada turma;
- Créditos e notas das disciplinas cursadas pelo alunos da turma;

Saída

Para cada turma, o programa deve fornecer:

- IRA individual de cada aluno de uma turma (IRA_i);
- IRA médio da turma ($\overline{IRA}$);
- Desvio-padrão do IRA na turma (σ);
- Maior e menor IRA da turma (IRA_{max} e IRA_{min});

5. BITS TROCADOS (OLIMPÍADA BRASILEIRA DE INFORMÁTICA)

Nome do arquivo fonte: bits.c, bits.cpp, bits.pas, bits.java, bits.py

As ilhas Weblands formam um reino independente nos mares do Pacífico. Como é um reino recente, a sociedade é muito influenciada pela informática. A moeda oficial é o Bit: existem notas de B$ 50,00, B$ 10,00, B$ 5,00e B$ 1,00. Você foi contratado(a) para ajudar na programação dos caixas automáticos de um grande banco das ilhas Weblands.

1. Tarefa

Os caixas eletrônicos das ilhas Weblands operam com todos os tipos de notas disponíveis, mantendo um estoque de cédulas para cada valor (B$ 50,00, B$ 10,00, B$ 5,00e B$ 1,00). Os clientes do banco utilizam os caixas eletrônicos para efetuar retiradas de um certo número inteiro de Bits.

Sua tarefa é escrever um programa que, dado o valor de Bits desejado pelo cliente, determine o número de cada uma das notas necessário para totalizar esse valor, de modo a minimizar a quantidade de cédulas entregues. Por exemplo, se o cliente deseja retirar B$ 50,00, basta entregar uma única nota de cinquenta Bits. Se o cliente deseja retirar B$ 72,00, é necessário entregar uma nota B$ 50,00, duas de B$ 10,00 e duas de B$ 1,00.

2. Entrada

A entrada é composta por vários conjuntos de teste. Cada conjunto de teste é composto por uma única linha, que contém um número inteiro positivo V, que indica o valor solicitado pelo cliente. O final da entrada é indicado por V=0.

Exemplo de Entrada:

1
72
0

3. Saída:

Para cada conjunto de teste de 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. Na segunda linha devem aparecer o conjunto de quatro inteiros I, J, K, L que representam o resultado encontrado pelo seu programa: I indica o número de cédulas de B$ 50,00, J indica o número de cédulas de B$ 10,00, K indica o número de cédulas de B$ 5,00 e L indica o número de cédulas de B$ 1,00. A terceira linha deve ser deixada em branco. A grafia mostrada no exemplo abaixo deve ser seguida rigorosamente.

Exemplo de Saída:

Teste 1
0 0 0 1
Teste 2
1 2 0 2

Esta saída corresponde ao exemplo da entrada mostrada no Exemplo de Entrada. Restrições: O≤V10000 (V=0 apenas para indicar o fim da entrada).

5. METEOROS (OLIMPÍADA BRASILEIRA DE INFORMÁTICA)

Nome do arquivo fonte: meteoros.c, meteoros.cpp, meteoros.pas, meteoros.java, meteoros.py

Em noites sem nuvens, pode-se muitas vezes observar pontos brilhantes no céu que se deslocam com grande velocidade, e em poucos segundos desaparecem de vista: são as chamadas estrelas cadentes ou meteoros. Meteoros são, na verdade, partículas de poeira de pequenas dimensões que, ao penetrar na atmosfera terrestre, queimam-se rapidamente (normalmente a uma altura entre 60 e 120 quilômetros). Se os meteoros são suficientemente grandes, podem não queimar completamente na atmosfera e dessa forma atingem a superfície terrestre: nesse caso são chamados de meteoritos.

Zé Felício é um fazendeiro que adora astronomia e descobriu um portal na Internet que fornece uma lista das posições onde caíram meteoritos. Com base nessa lista, e conhecendo a localização de sua fazenda, Zé Felício deseja saber quantos meteoritos caíram dentro de sua propriedade. Ele precisa da sua ajuda para escrever um programa de computador que fala essa verificação automaticamente.

1. Tarefa

São dados:

- Uma lista de pontos no plano cartesiano, onde cada ponto corresponde à posição onde caiu um meteorito;
- As coordenadas de um retângulo que delimita uma fazenda.

As linhas que delimitam a fazenda são paralelas aos eixos cartesianos. Sua tarefa é escrever um programa que determine quantos meteoritos caíram dentro da fazenda (incluindo meteoritos que caíram exatamente sobre as linhas que delimitam a fazenda).

2. Entrada

Seu programa deve ler vários conjuntos de testes. A primeira linha de um conjunto de testes contém quatro números inteiros X_1 , Y_1, X_2 e Y_2, onde (X_1 , Y_1) é a coordenada do canto superior esquerdo e (X_2 e Y_2) é a coordenada do canto inferior direito do retângulo que delimita a fazenda. A segunda linha contém um inteiro N, que indica o número de meteoritos. Seguem-se N linhas, cada uma contendo dois números inteiros X e Y, correspondendo às coordenadas de cada meteorito. O final da entrada é indicado por X_1=Y_1=X_2=Y_2=0.

Exemplo de Entrada

```
2 4 5 1
2
1 2
3 3
2 4 3 2
3
1 1
2 2
3 3
0 0 0 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 o número de meteoritos que caíram dentro da fazenda. 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
1
Teste 2
2

Esta saída corresponde ao exemplo de entrada acima.

4. Restrições

$0 \leq N \leq 10000$

$0 \leq X \leq 10000$

$0 \leq Y \leq 10000$

$0 \leq X_1 \leq X_2 \leq 10000$

$0 \leq Y_1 \leq Y_2 \leq 10000$

6. FALANDO EM LÍNGUAS (GOOGLE CODE JAM)

Nome do arquivo fonte: linguas.c, linguas.cpp, linguas.pas, linguas.java, linguas.py

Problema

Temor a melhor língua possível aqui na Google, chamado Googlerese. Para traduzir um texto para Googlerese, tomamos qualquer mensagem e substituídmos cada letra em inglês por outra letra em inglês. Esse mapeamento é de um para um, o que significa que as mesmas entradas de letras sempre são substituídas pelas mesmas letras de saída e que entradas de letras diferentes são sempre substituídas por saídas de letras diferentes. Uma letra pode ser substituída por ela mesma. Espaços permanecem como estão.

Por exemplo (e aqui segue uma dica!), o nosso impressionante algoritmode tradução inclui os seguintes três mapeamentos: 'a' -> 'y', 'o' -> 'e' e 'z'->'q'. O que significa que "a zoo" será traduzido para "y qee".

Googlerese é baseado no melhor mapeamento de substituição possível e nós nunca iremos mudá-lo. Será sempre o mesmo. Em todo caso de teste. Nós não contaremos o resto do nosso mapeamento porque isso tornaria o problema muito fácil, mas há alguns axemplos adiante que poderão ajudá-lo.

Dados alguns textos em Googlerese, você poderia traduzi-los de volta para textos normais?

Resolvendo este problema

Geralmente, os problemas do Google Code Jam têm 1 entrada pequena e 1 entrada grande. Este problema possui apenas uma entrada pequena. Uma vez você tenha resolvido esta entrada pequena, você resolveu este problema.

Entrada

A primeira linha da entrada fornece o número de casos de testes, T. T testes seguem, um por linha.

Cada linha consiste de uma string G em Googlerese, construída por uma ou mais palavras contendo letras de 'a' - 'z'. Haverá exatamente um caractere espaço (' ') entre duas palavras consecutivas e não haverão espaços no início nem no fim das linhas.

Saída

Para cada caso de teste, haverá uma linha de saída contendo "Caso #X: S" onde X é o número do caso e S é a string traduzida a partir de G em Googlerese.

Limites

$1 \leq T \leq 30$

G contém pelo menos 100 caracteres

Nenhum texto é garantido ser válido em inglês.

Exemplo:

Entrada

3

ejp mysljylc kd kxveddknmc re jsicpdrysi

rbcpc ypc rtcsra dkh wyfrepkym veddknkmkrkcd

de kr kd eoya kw aej tysr re ujdr lkgc jv

Saída

Caso #1: our language is impossible to understand

Caso #2: there are twenty six factorial possibilities

Caso #3: so it is okay if you want to just give up

7. EXTENSÕES DE TOMADAS (OLIMPÍADA NÓRDICA DE PROGRAMAÇÃO)

Nome do arquivo fonte: tomadas.c, tomadas.cpp, tomadas.pas, tomadas.java, tomadas.py

Roy acabou de se mudar para um novo apartamento. Bem, realmente, o apartamento em si não é muito novo, datando dos dias em que as pessoas não tinham eletricidade em suas casas. Por causa disso, o apartamento de Roy tem apenas uma única tomada na parede, então Roy pode apenas um de seus aparelhos elétricos por vez.

Roy gosta de assistir TV enquanto trabalha no computador, e ouvir se com (no máximo volume) enquanto ele trabalha, então usar uma única tomada não é uma opção. Realmente, ele que ter todos os seus aparelhos conectados à rede elétrica ao mesmo tempo. A resposta, é claro, são as extensões de tomada, e Roy tem alguns modelos velhos destes que usava em seu antigo apartamento. Entretanto, seu antigo apartamento tinha muito mais tomadas nas paredes, então ele não está certo se suas extensões fornecerão, agora, tomadas suficientes.

Sua tarefa é ajudar Roy a contar quantos aparelhos ele pode fornecer com eletricidade, dado um conjunto de extensões. Note que, sem qualquer extensão, Roy pode ligar um único aparelho na tomada da parede. Lembre-se também, de que devemos manter o filtro de linha ligado à rede elétrica para ser usado.

Especificações de Entrada

A entrada vai começar com um único inteiro $1 \leq N \leq 20$, indicando o número de testes que se seguem. Então seguem-se as N linhas. Cada caso de teste começa com um inteiro $1 \leq K \leq 20$, indicando o número de extensões em cada caso de teste. Então seguem, na mesma linha, K inteiros separados por um único espaço, O_1 O_2 ... O_K, onde $2 \leq O_i \leq 10$, indicando o número de tomadas de cada extensão.

Especificações de Saída

Deve ser produzida uma linha para cada caso de teste, com o número máximo de aparelhos que podem ser ligados.

Exemplo de entrada

```
3
3 2 3 4
10 4 4 4 4 4 4 4 4 4
4 10 10 10 10
```

Exemplo de Saída

```
7
31
37
```

8. FATORIAL

Nome do arquivo fonte: fatorial.c, fatorial.cpp, fatorial.pas, tomadas.java, fatorial.py

A operação fatorial é definida para naturais por meio da seguinte expressão

$$n! = \prod_{k=1}^{n} k \, , \forall n \in N$$

Assim, $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$. Vale ressaltar que $0! = 1$. A notação $n!$ Para o fatorial foi introduzida em 1808 por Cristian Kramp.

Especificações de Entrada

A entrada deve ser iniciada com os *i* casos de teste a serem executadas, seguidas por *i* linhas, onde cada linha representa uma entrada. O programa deve ser encerrado quando o usuário digitar 0 para o número de casos de teste.

Especificações de Saída

Deve ser produzida, para cada valor *n* entrado, uma resposta indicando o valor de *n!*, sendo que cada bateria deve ser precedida pela expressão "Teste i", onde i deve indicar o número do teste. Os casos de teste devem ser separados por uma linha em branco.

Exemplo de entrada

```
3
0
4
5

2
2
3

0
```

Exemplo de Saída

```
Teste 1:
1
4
120

Teste 2:
2
6
```

9. SISTEMA NUMÉRICO BINÁRIO

Nome do arquivo fonte: binário.c, binário.cpp, binário.pas, binário.java, binário.py

Sistemas digitais, incluindo computador, são sistemas baseados em sistema de numeração binária, ou seja, possui apenas os dígitos 0 e 1, indicando estados de baixa e alta tensão nos circuitos eletrônicos. Daí o conceito de bit (***bi***nary digi***t***). Um bit, representa um estado, 0 ou 1. Um byte é um conjunto de 8 bits. Sua tarefa é fazer um programa que, dado uma entrada, realize a conversão binária-decimal ou vice-versa.

A. A conversão binário para decimal se dá da seguinte forma. Dado um número em binário escrito da seguinte forma:

$$b_n b_{n-1} \dots b_3 b_2 b_1 b_0$$

Seu valor em base decimal é dado por:

$$x = \sum_{k=0}^{n} 2^k b_k$$

B. A conversão de decimal para binário se dá da seguinte forma:

- Dividir o número decimal por 2. Caso o resultado seja exato, aquela divisão terá resto 0 (zero), se não for exato terá resto 1 (um). Esse valor deve ser anotado da direita para a esquerda.
- Deve-se dividir o número até que o quociente da divisão seja igual a 0 (zero).
- Observe que o resultado é obtido juntando o resultado da última para a primeira divisão, ou seja, de baixo para cima.

Especificações de entrada:

A entrada deve ser iniciada com um valor i indicando os i casos de teste que devem se suceder para cada teste. As i linhas seguintes devem indicar, individualmente, as os valores com os quais se deseja converter sendo sucedido pela letra b indicando que o valor se encontra em base binária ou d indicando que o sistema se encontra em base decimal. O programa deve se encerrar indicando valor 0 para o número de casos de testes.

Especificações de Saída

Cada linha deve produzir uma resposta correspondente a uma entrada de uma bateria de testes. Cada bateria de testes deve produzir uma linha inicial indicando "Teste i". Os casos de teste devem ser separados por uma linha em branco.

Exemplo de Entrada:

2
1985 d
11111000001 b
0

Exemplo de Saída:

Teste 1
11111000001
1985

10. TRUQUE MÁGICO (GOOGLE CODE JAM 2014)

Nome do arquivo fonte: magico.c, magico.cpp, magico.pas, magico.java, magico.py

Problema

Recentemente você foi para um show de mágica. Você ficou muito impressionado com um dos truques, então você decidiu tentar descobrir qual o segredo por trás dele.

O mágico começa organizando 16 cartas em um quadrado: 4 linhas de cartas com 4 cartas cada. Cada carta tem um número diferente de 1 a 16 escrito no lado que está sendo mostrado. Em seguida, o mágico pergunta a um voluntário para escolher uma carta e dizer em que linha a carta está.

Finalmente o mágico organiza as 16 cartas em um quadrado novamente, possivelmente em uma ordem diferente. Mais uma vez, ele pede para o voluntário indicar em que linha está a carta. Com apenas as respostas destas duas perguntas, o mágico determina corretamente qual carta o voluntário escolheu. Incrível, não?

Você decide escrever um programa para ajudar a entender a técnica do mágico. O programa receberá dois arranjos de cartas e as respostas do voluntário para as duas questões: o número da linha da carta selecionada e o número da linha da carta selecionada no segundo arranjo. As linhas são numeradas de 1 a 4 de cima a baixo.

Seu programa deve determinar qual carta o voluntário escolheu, ou se há mais de uma carta escolhida pelo mágico (o mágico fez um mau trabalho) ou se não há uma carta consistente com as respostas do voluntário (o voluntário trapaceou).

Resolvendo o problema

Geralmente, o Google Code Jam possui uma entrada pequena e uma entrada grande. Este problema possui apenas uma entrada pequena. Uma vez resolvido o problema para a entrada pequena, você terminou de resolver o problema.

Entrada

A primeira linha da entrada fornece o número de casos de teste T. Seguem-se então T casos de teste. Cada teste inicia com uma linha contendo um inteiro: a resposta para a primeira pergunta. As próximas 4 linhas representam o primeiro arranjo de cartas: cada uma contém 4 inteiros, separados por um único espaço. A linha seguinte contém a resposta para a segunda pergunta, e as quatro linhas seguintes contém o segundo arranjo no mesmo formato.

Saída

Para cada caso de teste, a linha de saída um contém "Caso #x: y", onde xdeve ser o número do caso de teste (começando por 1).

Se há uma única possível carta que o voluntário escolheu, y deve ser o número da carta. Se há múltiplas cartas que o voluntário possa ter escolhido, y deve ser "Mágico Ruim", sem as aspas. Se não há cartas consistentes com as respostas do voluntário, y deve ser "Voluntário Trapaceou!", sem as asas. O texto deve ser estritamente correto, então considere copiar e colar daqui.

Limites

$1 \leq T \leq 100$.

$1 \leq$ ambas as respostas ≤ 4.

Cada número de 1 a 16 deve aparecer exatamente uma vez no arranjo

Exemplo

Entrada

```
3
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
3
1 2 5 4
3 11 6 15
9 10 7 12
13 14 8 16
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
2
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
3
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Saída

```
Caso #1: 7
Caso #2: Mágico Ruim
Caso #3: Voluntário Trapaceou!
```

11. PAR OU ÍMPAR

Nome do arquivo fonte: paridade.c, paridade.cpp, paridade.pas, paridade.java, paridade.py

Problema

Escreva um programa que determine a paridade de um número. O programa deve realizar diversos casos de testes.

Entrada

A primeira linha deve conter o número *n* de testes a serem realizados dentro da bateria de testes em questão. As demais *n* linhas devem conter, cada uma os *n* números que devem ser verificadas as paridades. O programa é encerrado quando *n = 0*.

Saída

Para cada caso de teste, deve ser mostrada uma linha inicial mostrando "Teste #I", sem aspas, onde I representa o número da bateria de testes. As N linhas seguintes devem indicar "Eh par", caso o número seja par, ou "Eh impar", caso seja ímpar. Caso valor seja zero, deve-se mostrar "Eh zero".

Exemplo

Entrada
2
1
3
3
2
5
6
0

Saída
Teste #1
Eh impar
Eh impar

Teste #2
Eh par
Eh impar
Eh par

12. PRIMOS

Nome do arquivo fonte: primos.c, primos.cpp, primos.pas, primos.java, primos.py

Problema

Denomina-se número primo aquele que é divisível apenas por 1 e ele mesmo. A sua tarefa é escrever um programa que determine se um dado número é primo ou não.

Entrada

A primeira linha deve conter o número *n* de testes a serem realizados dentro da bateria de testes em questão. As demais *n* linhas devem conter, cada uma os *n* números que devem ser verificados se são primos ou não. O programa é encerrado quando *n = 0*.

Saída

Para cada caso de teste, deve ser mostrada uma linha inicial mostrando "Teste #I", sem aspas, onde I representa o número da bateria de testes. As N linhas seguintes devem indicar "Eh primo", caso o número seja primo, ou "Eh composto", caso seja composto. Caso valor seja zero, deve-se mostrar "Eh zero".

Exemplo

Entrada
2
5
7
3
2
4
9

Saída
Teste #1
Eh primo
Eh primo

Teste #2
Eh primo
Eh composto
Eh primo

13. PALÍNDROMOS

Nome do arquivo fonte: palindromo.c, palindromo.cpp, palindromo.pas, palindromo.java, palindromo.py

Problema

Denomina-se palíndromo uma palavra que pode ser lida tanto em ordem convencional como de trás-para-frente. Exemplo: Ana. Sua tarefa é determinar se uma dada palavra é palíndromo ou não.

Entrada

A primeira linha deve conter o número *n* de testes a serem realizados dentro da bateria de testes em questão. As demais *n* linhas devem conter, cada uma as palavras que devem ser verificadas se são palíndromos ou não. O programa é encerrado quando *n = 0*.

Saída

Para cada caso de teste, deve ser mostrada uma linha inicial mostrando "Teste #I", sem aspas, onde I representa o número da bateria de testes. As N linhas seguintes devem indicar "Eh palindromo", caso a palavra seja palíndromo, ou "Nao eh palindromo", caso contrário.

Exemplo

Entrada
4
Asa
Carro
Ana
Casa
0

Saída
Teste #1
Eh palíndromo
Nao eh palindromo
Eh palíndromo
Nao eh palindromo

14. AREAS

Nome do arquivo fonte: areas.c, areas.cpp, areas.pas, areas.java, areas.py

Problema

Escreva um programa que determine a área de um triângulo, retângulo, trapézio ou circunferência.

Entrada

O programa é composto por um vários casos de teste. O usuário deve obedecer à seguinte entrada:

- T b h, caso se deseje calcular a área de um triângulo, sendo b o tamanho da base e h o tamanho da altura;
- R b h, caso se deseje calcular a área de um retângulo, sendo b o tamanho da base e h o tamanho da altura;
- A B b h, caso se deseje calcular a área de um trapézio, sendo B o tamanho da base maior, b o tamanho da base menor e h o tamanho da altura;
- C r, caso se deseje calcular a área de uma circunferência, sendo r o tamanho do raio.

O programa é encerrado com entrada 0.

Saída

O programa deve fornecer a área da figura geométrica desejada, obedecendo a seguinte formatação: Caso #1: Valor_da_área.

Exemplo

Entrada
R 2.5 2
T 2 2
0

Saída
Caso #1: 5
Caso #2: 2

15. OPERADORES RELACIONAIS (LISTA DE EXERCÍCIOS DA UNIVERSIDADE DE VALLADOLLID)

Nome do arquivo fonte: operadores.c, operadores.cpp, operadores.pas, operadores.java, operadores.py

Alguns operadores verificam a relação entre dois valores. Tais operadores são denominados de operadores relacionais. Dados dois valores numéricos, sua tarefa é apenas encontrar a relação entre ambos. Se (i) o primeiro é maior que o segundo (ii) o primeiro é menor que o segundo (iii) o primeiro é igual ao segundo.

Entrada

A primeira linha da entrada é um inteiro t (t<15) que denota quantos são os conjuntos de entradas. Cada uma das t linhas seguintes contém dois inteiros a e b (|a|,|b| < 1000000001).

Saída

Cada linha de entrada é seguida imediatamente por uma linha de saída. Esta linha contém qualquer um dos operadores relacionais '>','<' ou '=', o qual contém a relação que é apropriada para os dois números dados.

Exemplo de Entrada

```
3
10 20
20 10
10 10
```

Exemplo de Saída

```
<
>
=
```

QUESTÃO 16. SOMA DE ÍMPARES (LISTA DE EXERCÍCIOS DA UNIVERSIDADE DE VALLADOLLID)

Nome do arquivo fonte: somaimpares.c, somaimpares.cpp, somaimpares.pas, somaimpares.java, somaimpares.py

Dado um intervalo [a,b], você deve encontrar a soma de todos os inteiros ímpares deste intervalo.

Entrada

Haverá múltiplos casos de teste. A primeira linha da entrada deve fornecer o número de casos de teste ($1 \leq T \leq 100$). Então seguem os T casos de teste. Cada teste consiste de 2 inteiros a e b ($0 \leq a \leq b \leq 100$) em duas linhas separadas.

Exemplo de Entrada

2
1
5
3
5

Exemplo de Saída

Caso 1: 9
Caso 2: 8

QUESTÃO 17. DISTÂNCIA ENTRE DOIS PONTOS

Nome do arquivo fonte: distancia.c, distancia.cpp, distancia.pas, distancia.java, distancia.py

Dados dois pontos P(x,y) e Q(x',y'), escreva um programa que calcule a distância entre os pontos P e Q. O programa deve ser composto por vários casos de teste, sabendo que $|PQ| = \sqrt{(x - x')^2 + (y - y')^2}$.

Entrada

Haverá múltiplos casos de teste. A primeira linha da entrada deve fornecer o número de casos de teste. Então seguem os T casos de teste. Cada teste consiste de 4 inteiros a serem fornecidos em uma única linha separados por um espaço único *x y x' y'*.

Saída

Para cada caso de teste, deve-se imprimir Caso #I: Resposta, onde I indica o número do caso de teste.

Exemplo de Entrada

1
0 0 4 4

Exemplo de Saída

Caso #1: 5.65

QUESTÃO 18. RAÍZES DE UMA EQUAÇÃO DO SEGUNDO GRAU

Nome do arquivo fonte: bhaskara.c, bhaskara.cpp, bhaskara.pas, bhaskara.java, bhaskara.py

Dada uma equação do segundo grau do tipo $P(x) = ax^2 + bx + c = 0$, onde os coeficientes $a, b, c \in R, a \neq 0$ são fornecidos pelo usuário e sabendo que a solução para esta equação foi proposta pelo matemático indiano Bhaskara é dada pela relação:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}, \Delta = b^2 - 4ac \geq 0$$

Entrada

Haverá múltiplos casos de teste. A primeira linha da entrada deve fornecer o número de casos de teste. Então seguem os T casos de teste. Cada teste consiste de 3 números reais que representam respectivamente os coeficientes a, b, c.

Saída

Para cada caso de teste, o programa deve fornece uma saída do seguinte tipo: Caso #1: x1=valor x2=valor. Caso o coeficiente A seja nulo, deve-se imprimir, "Coeficientes inválidos", sem aspas, caso $\Delta < 0$, deve-se imprimir, "As raízes são complexas."

Exemplo de Entrada

3
0 1 2
111
121

Exemplo de Saída

Caso #1: Coeficientes inválidos
Caso #2: As raízes sao complexas
Caso #3: x1= -1 x2= -1

QUESTÃO 19. TRIÂNGULOS

Nome do arquivo fonte: triangulos.c, triangulos.cpp, triangulos.pas, triangulos.java, triangulos.py

Conhecidas as relações para formação de um triângulo, determine se, dadas as dimensões dos lados de um triângulo, o triângulo existe ou não.

Entrada

Este problema é composto por um único caso de teste. É fornecida apenas uma linha com as dimensões de cada um dos três lados separados apenas por um único espaço entre si.

Saída

O programa deve simplesmente mostrar a mensagem "O dado triângulo existe" caso o triângulo seja possível, ou "O dado triangulo não existe", caso contrário, sem as aspas.

Exemplo de Entrada

1 1 1

Exemplo de Saída

O dado triangulo não existe

QUESTÃO 20. CONVERSOR DE TEMPERATURAS E DE MOEDAS

Nome do arquivo fonte: conversor.c, conversor.cpp, conversor.pas, conversor.java, conversor.py

O Jovem Nícolas gosta de assistir BBC quando está em casa programando, pois gosta de treinar seu inglês. Entretanto, quando mostra a previsão do tempo ele se perde, uma vez que, embora saiba a expressão que converte Celsius para Fahrenheit, ele detesta fazer conta de cabeça (afinal, foi para isso que inventaram os computadores!).

O grau fahrenheit (símbolo: °F) é uma escala de temperatura proposta por Daniel Gabriel Fahrenheit em 1724. Nesta escala, o ponto de fusão da água (0 °C) é de 32 °F. Trata-se de uma escala muito usada nos países colonizados por ingleses (p. ex.: Estados Unidos). Sabendo que:

$$\frac{T_C}{5} = \frac{T_F - 32}{9}$$

Onde T_C é a temperatura na escala Celsius e T_F é a temperatura em Fahrenheit, e que a cotação da libra esterlina enquanto ele cria esta primorosa prova está em R$ 4,51, crie um programa que converta a temperatura de Fahrenheit para Celsius e de Real para Libras.

Entrada

A entrada deve seguir o formato *x y*. Onde *x* representa a temperatura em Fahrenheit e *y* o valor em Real. O programa é encerrado com entrada 0 0.

Saída

O programa deve mostrar os valores convertidos em uma única linha, para cada teste.

Exemplo de Entrada

0 100.00
5 0.00
100 4.51
0 0

Exemplo de Saída

32 22.17
41 0.00
212 1.00

QUESTÃO 21. ASCII

Nome do arquivo fonte: ascii.c, ascii.cpp, ascii.pas, ascii.java, ascii.py

A tabela ASCII - do inglês *American Standard Code for Information Interchange*; "Código Padrão Americano para o Intercâmbio de Informação" (Figura 6) mostra a representação de diversos caracteres (incluindo números, letras maiúsculas e minúsculas e caracteres não imprimíveis) nas bases binário, octal, decimal e hexadecimal. Considerando que computadores são sistemas digitais e trabalham essencialmente com aritmética, a tabela ASCII é especialmente útil para programadores.

Bin	Oct	Dec	Hex	Sinal
0010 0000	040	32	20	(espaço)
0010 0001	041	33	21	!
0010 0010	042	34	22	"
0010 0011	043	35	23	#
0010 0100	044	36	24	$
0010 0101	045	37	25	%
0010 0110	046	38	26	&
0010 0111	047	39	27	'
0010 1000	050	40	28	(
0010 1001	051	41	29	)
0010 1010	052	42	2A	*
0010 1011	053	43	2B	+
0010 1100	054	44	2C	,
0010 1101	055	45	2D	-
0010 1110	056	46	2E	.
0010 1111	057	47	2F	/
0011 0000	060	48	30	0
0011 0001	061	49	31	1
0011 0010	062	50	32	2
0011 0011	063	51	33	3
0011 0100	064	52	34	4
0011 0101	065	53	35	5
0011 0110	066	54	36	6
0011 0111	067	55	37	7
0011 1000	070	56	38	8
0011 1001	071	57	39	9
0011 1010	072	58	3A	:
0011 1011	073	59	3B	;
0011 1100	074	60	3C	<
0011 1101	075	61	3D	=
0011 1110	076	62	3E	>
0011 1111	077	63	3F	?

Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F
0100 0111	107	71	47	G
0100 1000	110	72	48	H
0100 1001	111	73	49	I
0100 1010	112	74	4A	J
0100 1011	113	75	4B	K
0100 1100	114	76	4C	L
0100 1101	115	77	4D	M
0100 1110	116	78	4E	N
0100 1111	117	79	4F	O
0101 0000	120	80	50	P
0101 0001	121	81	51	Q
0101 0010	122	82	52	R
0101 0011	123	83	53	S
0101 0100	124	84	54	T
0101 0101	125	85	55	U
0101 0110	126	86	56	V
0101 0111	127	87	57	W
0101 1000	130	88	58	X
0101 1001	131	89	59	Y
0101 1010	132	90	5A	Z
0101 1011	133	91	5B	[
0101 1100	134	92	5C	\
0101 1101	135	93	5D	]
0101 1110	136	94	5E	^
0101 1111	137	95	5F	_

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f
0110 0111	147	103	67	g
0110 1000	150	104	68	h
0110 1001	151	105	69	i
0110 1010	152	106	6A	j
0110 1011	153	107	6B	k
0110 1100	154	108	6C	l
0110 1101	155	109	6D	m
0110 1110	156	110	6E	n
0110 1111	157	111	6F	o
0111 0000	160	112	70	p
0111 0001	161	113	71	q
0111 0010	162	114	72	r
0111 0011	163	115	73	s
0111 0100	164	116	74	t
0111 0101	165	117	75	u
0111 0110	166	118	76	v
0111 0111	167	119	77	w
0111 1000	170	120	78	x
0111 1001	171	121	79	y
0111 1010	172	122	7A	z
0111 1011	173	123	7B	{
0111 1100	174	124	7C	|
0111 1101	175	125	7D	}
0111 1110	176	126	7E	~

Figura 6. Tabela ASCII. Fonte: Wikipedia.

Nícolas é um jovem programador, mestrando em Engenharia de Teleinformática da Universidade Federal do Ceará e que está precisando da tabela ASCII para fazer a prova que irá aplicar para os seus maravilhosos alunos no fim de semana em um renomado colégio na cidade onde mora. Sabendo que a preguiça é a mãe do progresso, e portanto o mesmo se recusa a se levantar de sua cama para pegar um livro que está no armário bem ao seu lado, ele resolve consultar a Wikipedia. Entretanto, o mesmo faz uso do serviço de internet móvel 3G de uma operadora de telefonia. Mesmo este jovem e liso bolsista da CAPES pagando religiosamente a

conta de R99,90 pelo serviço, volta e meia esta operadora o deixa na mão, o deixando sem internet nos momentos que mais precisa ou fazendo com que os bits venham de jegue...

Tarefa

Você, como bom aluno que é, mostre que tem dado a devida atenção em suas aulas semanais e matinais de sábado preparatórias para a Olimpíada Brasileira de Informática, faça um programa que, dado um determinado caractere, mostre seu valor em decimal na tela.

Entrada

Este problema é composto por diversas entradas. Cada linha deve receber unicamente um caracter. O programa é encerrado com um enter, sem entrada.

Saída

O programa deve simplesmente mostrar para cada caractere digitado, seu respectivo valor ASCII.

Exemplo de Entrada

m
Y
U
:

7

Exemplo de Saída

109
89
85
58
55