Problema A: Cara & Coroa

Arquivo fonte: `caracoroa.c, caracoroa.cpp, caracoroa.java ou caracoroa.pas`

Em função das atuais notícias de corrupção no futebol, os torcedores desconfiaram até mesmo da legitimidade do sorteio de lados (o tradicional "cara e coroa") realizado no inicio das partidas. Atendendo suas reclamações, os clubes solicitaram que fosse desenvolvida uma máquina que joga moedas para cima a fim de automatizar o sorteio no início das partidas do campeonato brasileiro. Porém, a Confederação Brasileira de Futebol (CBF) teme que até mesmo a máquina possa estar adulterada e pede a sua ajuda para verificar a confiabilidade do novo dispositivo.

Acredita-se que, se a máquina for "honesta", a quantidade de vezes sorteadas de 'caras' e 'coroas' deva ser igual ou muito próximo, segundo os princípios estatísticos.

Você foi contratado para desenvolver um programa que analise os arquivos de log gerados pelos sorteios da máquina, a fim de averiguar se um dos lados está tendendo a ser sorteado mais vezes. Para tal, o programa deve apresentar um relatório contendo a quantidade máxima de vezes repetidas de caras e de coroas ocorridas durante cada campeonato.

Os arquivos de log são armazenados por campeonato, ou seja, a cada campeonato são armazenados os resultados de todos os sorteios, bem como a quantidade de jogos realizados. Um arquivo de log pode conter dados de vários campeonatos diferentes. Assim, um arquivo de log é composto por números inteiros respeitando sempre a seguinte estrutura: na primeira linha, um número N indica a quantidade de jogos realizados naquele campeonato. A linha seguinte, contém uma sequência de 0´s e 1´s representando os resultados de cada sorteio, onde 0 representa cara e 1 representa coroa. É utilizado o valor N=0 para indicar o fim do arquivo de log.

Pede-se: Apresentar o máximo de ocorrências repetidas de caras e de coroas em cada campeonato.

Entrada

```
20
00111011110001110010
10
1111101110
0
```

Saída

```
3 caras e 4 coroas
1 caras e 5 coroas
```

Problema B: Árvore Estranha

Arquivo fonte: `arvore.c, arvore.cpp, arvore.java ou arvore.pas`

Uma Árvore Estranha, do inglês *Strange Tree* (S-Tree), sobre o conjunto de variáveis $X_n = \{X_1, \ldots, X_n\}$ é uma árvore binária completa representando uma função booleana $f : \{0,1\}^n \rightarrow \{0,1\}$. Cada um dos nós da S-tree tem uma profundidade, definida como o número de nós existentes no caminho da raiz ate o próprio nó menos 1 (portanto a raiz tem profundidade 0). A profundidade de qualquer nó em uma S-tree é, no máximo, n. Os nós com profundidade menor que n são chamados nós não terminais, cada um tendo dois filhos: o filho da direita e o filho da esquerda. Cada nó não terminal é nominado com alguma variável X_i do conjunto de variáveis X_n. Todos os nós não terminais com a mesma profundidade são nominados com a mesma variável, e nós terminais com diferentes profundidades são nominados com variáveis diferentes. Então, existe uma variável unica X_0 correspondente a raiz, uma variável única X_1 correspondente aos nós com profundidade 1, e assim por diante. A sequência de variaveis $X_0, X_1, \ldots, X_{n-1}$ é chamada de ordenação das variáveis. Os nós que têm profundidade n são chamados nós terminais. Eles não tem filhos e são nominados com ou 0 ou 1. Note que a ordem das variáveis e a distribuição de 0's e 1's nos nós terminais são suficientes para descrever completamente uma S-tree.

Como constatado anteriormente, cada S-tree representa uma função booleana f. Se você tiver uma S-tree e os valores para as variáveis $X_1, \ldots, X_n$, entao é bastante simples calcular $f(X_1, \ldots, X_n)$: comece na raiz e repita os seguintes passos: se o nó em que você está é nominado por uma variavel X_i, entao dependendo do valor da variavel X_i ser 1 ou 0, você vai para o seu filho da direita ou da esquerda respectivamente. Assim que você atingir um nó terminal, a nominação deste nó é o valor da função.

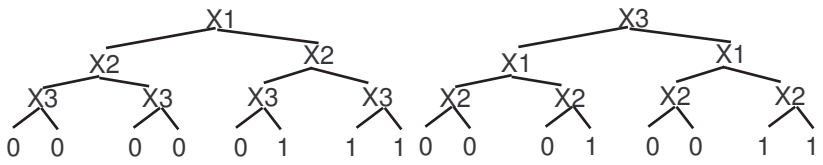

Figura 1: S-trees para a função X1 and (X2 or X3)

Na figura 1, as duas S-trees representam a mesma função booleana:

$$f(x1, x2, x3) = x1 \; and \; (x2 \; or \; x3)$$

Para a árvore da esquerda, a ordenação das variáveis é x1, x2, x3 e para a arvore da direita a ordenacao é x3, x1, x2. Os valores das variáveis x1, ..., xn, são dados como uma Associação de Valores de Variáveis (*Variable Value Assignment* - VVA) (x1=b1, x2=b2, ..., xn=bn) com b1, b2, ..., bn E 0,1. Por exemplo, (x1=1, x2=1, x3=0) seria uma VVA para n=3, resultando, para a simples função acima, no valor f(1,1,0)= 1 and (1 or 0) = 1.

Tarefa

Sua tarefa é escrever um programa que, dadas uma S-tree e alguns VVA's, calcule f(x1, ..., xn) como descrito acima.

Entrada

O arquivo de entrada contém a descrição de diversas S-trees com VVA's associados, os quais você deve processar. Cada descrição começa com uma linha contendo um inteiro apenas n (1¡=n¡=7), que é a profundidade da S-tree. Isto é seguido por uma linha descrevendo a ordenação das variáveis da S-tree. O formato da linha e $X_1, X_2, \ldots, X_n$. (Existirão exatamente n strings separadas por um espaço apenas). Portanto, para n=3 e a ordenação de variáveis X_3, X_1, X_2, esta linha seria:

$$X_3\ X_1\ X_2$$

Na linha seguinte é dada a distribuição de 0's e 1's nós nos terminais. Existirão exatamente 2^n caracteres (cada um deles pode ser 0 ou 1), seguidos por um caracter de *new-line* (fim de linha). Os caracteres são dados na ordem em que eles aparecem na S-tree, o primeiro caracter corresponde ao nó terminal mais a esquerda e o último corresponde ao nó terminal mais a direita.

A próxima linha contem um inteiro m apenas simbolizando o número de VVA's, seguido por m linhas descrevendo estas VVA's. Cada uma das m linhas contêm exatamente n caracteres (cada um dos quais pode ser 0 ou 1), seguidos por um caracter de *new-line*. Independente da ordenação de variáveis da S-tree, o primeiro caracter sempre descreve o valor de X_1, o segundo descreve o valor de X_2 e assim por diante, Então, a linha

110

corresponde ao VVA ($X_1 = 1, X_2 = 1, X_3 = 0$).

A entrada é terminada com um caso de teste começando com n=0. Este caso de teste não deve ser processado.

Exemplo de Entrada

```
3
x1 x2 x3
00000111
4
000
010
111
110
3
x3 x1 x2
00010011
4
000
010
111
110
0
```

Saída

Para cada S-tree deve ser impresso a linha "`S-Tree #j`", onde j é o número da S-tree. Então imprima uma linha que contém o valor de f($X_1, X_2, \ldots, X_n$) para cada uma das m VVA's dadas, onde f é a função definida pela S-tree. Imprima uma linha em branco após cada caso de teste.

Exemplo de Saída

```
S-Tree #1
0011

S-Tree #2
0011
```

Problema C: Números Palíndromos

Arquivo fonte: `palindromo.c, palindromo.cpp, palindromo.java ou palindromo.pas`

Dizemos que um número é um palíndromo se ele é o mesmo quando lido da esquerda para a direita quanto da direita para a esquerda. Por exemplo, o número 75457 é um palíndromo.

É claro que a propriedade depende da base na qual o número está representado. O número 17 não é um palíndromo se representado na base 10, porém sua representação na base 2 (10001) é um palíndromo.

O objetivo deste problema é verificar se um conjunto de números são palíndromos em qualquer base de 2 a 16.

Tarefa

Sua tarefa é escrever um programa que, dado um conjunto de números diga em quais bases, de 2 a 16, eles são palíndromos.

Entrada

A entrada é composta por diversos números inteiros. Cada número, de 0 a 50000, é dado na base decimal, em uma linha separada. A entrada termina com um zero.

Exemplo de Entrada

```
7
19
0
```

Saída

O seu programa deve imprimir a mensagem "Number i is a palindrom in basis", onde i é o número dado, seguido pelas bases nas quais o número é um palíndromo. Se o número não é um palíndromo em nennhuma base entre 2 e 16 seu programa deve imprimir a mensagem "Number i is not a palindrom".

Exemplo de Saída

```
Number 17 is a palindrom in basis 2 4 16
Number 19 is not a palindrom
```

Problema D: Catacumbas

Arquivo fonte: `catacumba.c, catacumba.cpp, catacumba.java ou catacumba.pas`

Nas catacumbas de um antigo castelo na Escócia, existem inúmeras passagens secretas lacradas por enormes e pesadas pedras de diversos formatos. Ao se tentar levantar um pedra, um mecanismo de segurança é ativado e flechas envenenadas são disparadas matando, quase que instantaneamente, quem estiver no local. A única forma segura de abrir essas passagens é levantar as pedras lenta e cuidadosamente, conectando uma corda à pedra no exato ponto correspondente a seu centro de gravidade, de forma a evitar que a pedra balance e acione o dispositivo de segurança. Estão disponíveis no local, alguns metros de corda e um guindaste rústico.

As pedras possuem sempre um formato poligonal e peso igualmente distribuído por todos seus pontos. Sua tarefa é encontrar esse centro de gravidade para um dado polígono.

Entrada

A entrada consiste de T casos de teste fornecido na primeira linha do arquivo de entrada. Cada caso de teste começa com uma linha contendo um único número inteiro N ($3 \leq N \leq 1000000$) indicando o número de pontos que formam um polígono. A seguir tem-se as N linhas, cada qual contendo dois números inteiros X_i e Y_i ($|X_i|, |Y_i| \leq 20000$). Este números são as coordenadas do *i-ésimo* ponto. Quando se conecta os pontos na ordem fornecida, tem-se o polígono. Pode-se assumir que os lados nunca se interceptam (exceto para lados vizinhos) e nunca se cruzam. A área do polígono nunca é zero.

Exemplo de Entrada

```
2
4
5 0
0 5
-5 0
0 -5
4
1 1
11 1
11 11
1 11
```

Saída

Imprima exatamente uma linha para cada caso de teste. A linha deve conter exatamente dois números separados por um caracter de espaço. Estes números são as coordenadas do centro de gravidade. Arredonde as coordenadas para o número mais próximo com exatamente duas casas decimais (por exemplo: 0.005 deve ser arredondado para 0.01). Perceba que o centro de gravidade pode estar fora do polígono (no caso de polígonos côncavos). Se isso acontecer, imprima o centro assim mesmo.

Exemplo de Saída

```
0.00 0.00
6.00 6.00
```

Problema E: Satélite Espião

Arquivo fonte: `satelite.c, satelite.cpp, satelite.java ou satelite.pas`

A agência espacial brasileira em parceira com o exército está desenvolvendo um novíssimo satélite espião, com o objetivo de detectar do espaço a localização de bases militares inimigas. Este satélite possui uma câmera de vídeo especial capaz de medir variações de temperatura na superfície do planeta, que são depois transformadas em imagens em tons de cinza (cinza escuro representa regiões frias e cinza claro as regiões quentes).

Como sabe-se, as bases militares secretas tem seu próprio gerador de energia que, vistos por essa câmera, apresentariam cores bem distintas das áreas vizinhas (mais frias). Criar um sistema capaz de identificar essas regiões é de fundamental importância para a inteligência militar.

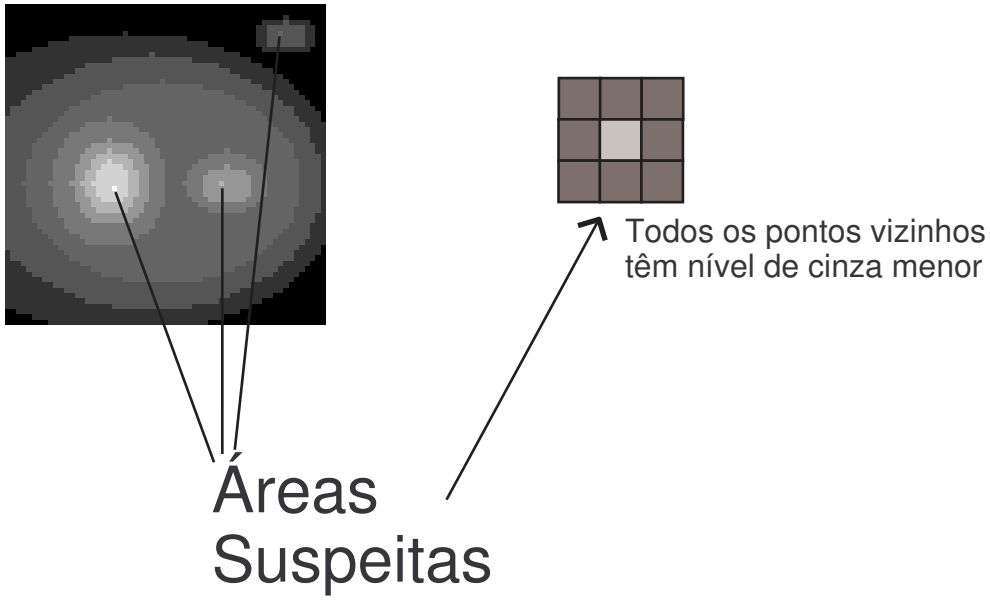

Figura 2: Detecção de áreas suspeitas a partir de fotos de satélite

Seu trabalho como espião recém contratado pelo governo brasileiro é desenvolver um programa capaz de, a partir de uma imagem desse satélite, encontrar todas as prováveis regiões das bases inimigas (conforme apresentado na figura 2). Como as imagens do satélite são imagens em tons de cinza, elas são representadas como uma matriz quadrada (tamanho NxN) de valores inteiros ($0 \leq \ Nivel\_Cinza \ \leq 255$). Os pontos considerados suspeitos têm um nível de cinza maior que todos os seus vizinhos imediatos (geralmente 8 pontos, exceto para os pontos na borda da imagem).

Tarefa

Criar um programa que analise várias imagens de satélite e apresente, em cada imagem, quantos são e quais as coordenadas de cada área suspeita.

Entrada

Um arquivo de entrada contém várias imagens de satélite podendo ser, inclusive, imagens de tamanhos (resoluções) distintos. A estrutura desse arquivo é composto por: na primeira linha, um

número inteiro N que representa o tamanho da imagem (largura e altura iguais a N). As próximas N linhas apresentam N colunas cada contendo números inteiros que correspondem às cores de cada ponto na imagem (em níveis de cinza C).

Existe uma linha em branco separando duas imagens. A primeira linha e a primeira coluna de cada imagem tem índice 0 (zero) e representam o canto superior esquerdo.

Exemplo de Entrada

```
5
0 0 0 0 0
1 10 15 15 20
5 100 14 15 80
0 25 24 240 20
0 0 0 0 0

7
0 0 0 0 0 0 200
18 10 10 10 10 10 0
0 10 10 10 10 10 0
0 10 50 10 10 10 0
0 10 10 10 10 10 0
0 10 10 10 195 10 0
0 0 0 0 0 0 0
```

Saída

Apresentar na primeira linha o número da imagem analisada. Nas próximas B linhas as coordenadas da linha Y e coluna X (separadas por um caracter de espaço) de cada uma das B bases inimigas. Os relatórios de cada imagem devem ser separados por uma linha em branco.

Exemplo de Entrada

```
1
2 1
3 3

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