Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema A

Tiro ao alvo

Nome do arquivo fonte: tiro.**{c, cpp** ou **java}**

Um super-herói está tentando salvar o planeta e, para isso, precisa atingir um alvo a quilômetros de distância com sua rajada laser. Nosso herói é conhecido por sua extrema precisão. Acontece que os super-vilões colocaram obstáculos entre ele e seu alvo de forma que apenas de tempos em tempos é possível atirar por frestas em cada obstáculo. Como exemplo, o obstáculo de frequência 5 só pode ser atravessado pelo laser a cada 5 segundos, nos instantes 5s, 10s, 15s e assim por diante.

Para piorar a situação, os super-vilões colocaram vários obstáculos um atrás do outro, de forma que só é possível atingir o alvo quando é possível que a rajada laser do herói passe pelas frestas de todos os obstáculos, sem exceção, ao mesmo tempo. Como outro exemplo, caso haja dois obstáculos, um com frequência 2 e outro com frequência 3, então só é possível atingir o alvo no instante 6 segundos (claro, também será possível nos instantes 12s, 18s e assim por diante).

Dadas as frequências de vários obstáculos que estão dispostos entre nosso herói e seu alvo, sua tarefa é informar qual o menor instante de tempo em que é possível atingir o alvo.

Entrada

A entrada é formada por vários casos de teste. Cada caso de testes é composto por duas linhas. A primeira linha possui um inteiro 0<= N <= 50 que indica o número de obstáculos entre o super-herói e o alvo. A segunda linha é composta por N números inteiros maiores que 0 e menores ou iguais a 30, indicando a frequência de cada obstáculo. **A entrada deve ser lida da entrada padrão.**

Saída

Para cada caso de testes da entrada você deve imprimir uma linha na saída contendo o menor instante de tempo em que é possível atingir o alvo. **A saída deve ser impressa na saída padrão.**

Exemplo de entrada	Respectiva saída
3 2 6 8 1 35 4 2 3 20 30 0	24 35 60

Autor: Eduardo Araújo, finalista da Maratona de Programação da SBC – Equipe UFMA Diott 2007.

Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema B

O Andarilho

Nome do arquivo fonte: anda.**{c, cpp** ou **java}**

Joãozinho é um garoto cheio de manias e que gosta muito de passear. Cada semana ele inventa uma mania nova. A mania dessa semana é andar de um lado para o outro (um pouco sem utilidade, certo, mas mania é mania, não dá para julgar!), sempre alternando N passos para esquerda e depois M passos para a direita e, assim sucessivamente, com novos (possivelmente diferentes) N passos para esquerda e outros M passos para a direita. Joãzinho pode tanto começar indo primeiro para esquerda como primeiro para a direita. Embora ele seja um garoto esperto, não é muito bom em matemática. Ficou interessado em saber qual a distância máxima que esse seu método o levaria do ponto inicial, mas não sabe muito bem como calcular isso.

Que tal ajudá-lo fazendo um programa para, dados o número de vezes que ele trocou de direção e todos os passos que ele deu indo para cada uma dessas direções, informá-lo qual a maior distância que ele ficou, em algum momento, do ponto de partida. Observe que Joãozinho pode começar tanto indo para a direita como indo para a esquerda, você sempre deve informar aquele lado em que for possível se distanciar mais do ponto inicial.

Entrada

A entrada é composta por vários casos de teste. Cada caso de testes é composto por duas linhas. A primeira linha contém um inteiro N, informando o número de vezes que Joãozinho trocou de direção. A seguir, segue a segunda linha da entrada, contendo N+1 inteiros, separados por espaço em branco, informando a quantidade de passos que ele deu, em cada direção. Observe que se Joãozinho começou indo para a direita, o primeiro número da entrada, p[1], informa que ele andou p[1] passos para a direita, o segundo que ele andou p[2] passos para a esquerda, e assim, sucessivamente, sempre alternando. Da mesma forma, ele também pode começar indo para a esquerda, depois para a direita, esquerda de novo, e assim sucessivamente. Quando N informado igual a zero, você deve terminar seu programa, ignorando essa entrada. **A entrada deve ser lida da entrada padrão.**

Saída

Para cada caso de teste da entrada, você deve imprimir um linha na saída contendo um inteiro informando qual a maior distância que Joãozinho pode ficar do ponto de partida, em algum momento durante o seu passeio.

Exemplo de entrada	Respectiva saída
3 10 2 4 5 2 5 5 5 4 10 5 10 5 10 0	12 5 20

Autor: Roberto Gerson.

Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema C

Contas Analíticas e Sintéticas

Nome do arquivo fonte: **contas.{c, cpp** ou **java}**

Os municípios têm que prestar contas mensalmente ao Estado dos débitos efetuados e créditos recebidos. Para tanto, o Estado recebe um arquivo com uma série de linhas, cada uma representando uma conta, com: i) o **código** da conta que será analisada; ii) o **tipo** da conta – se é Analítica "A" ou sintética "S"; e iii) os valores respectivamente de **crédito** e **débito,** o que só é apresentado para contas sintéticas.

Calcular o saldo das contas analíticas não é tão trivial, uma vez que apenas as contas sintéticas informam os valores de crédito e débito diretamente. Os valores para as contas analíticas devem ser calculados a partir da soma das contas imediatamente abaixo delas seguindo uma **hierarquia.** A hierarquia é definida assim: uma conta analítica de código X com D dígitos (excluindo seus zeros à esquerda e direita) tem seu total de créditos igual à soma de todos os créditos nas contas (analíticas ou sintéticas) com D+1 dígitos (novamente, excluindo zeros à esquerda ou direita) cujos D dígitos iniciais dessas contas são exatamente iguais aos D dígitos de X. A mesma definição também vale para a soma dos débitos. Por definição, uma conta analítica sem ninguém abaixo dela na hierarquia tem seus totais de crédito e débito iguais a ZERO (0,00).

Eis um exemplo com 7 contas: i) 1 A; ii) 11 A; iii) 111 A; iv) 1111 S 12.00 13.00; v) 1112 S 4.00 0.00; vi) 112 A; vii) 1121 S 11.00 15.00. A conta 111 é igual a soma das contas 1111 e da 1112 (111 tem 3 dígitos, portanto é a soma de todas as contas de exatamente 4 dígitos cujos primeiros 3 dígitos são 111). A conta 112 é a soma da conta 1121 (única cujos 3 primeiros dígitos são 112). A conta 11 é a soma das contas 111 e 112 (porque ambas têm seus dois primeiros dígitos iguais a 11). Enfim, a conta 1 é a soma da conta 11. Note que se não existisse a conta 1121, o total de créditos e débitos de 112, por definição, seria considerado igual a 0,00.

Acontece que as contas são informadas na entrada com exatamente 14 (quatorze) dígitos, apesar que nem todas as contas **TÊM** 14 dígitos. Para resolver esse problema, os dígitos faltantes aparecem na entrada com zeros. Isso significa que pode haver qualquer quantidade de zeros à esquerda ou à direita do código da conta – os quais devem ser ignorados - para completar 14 dígitos. A conta "1", por exemplo, pode aparecer na entrada como 00000000000001 ou 00000000000100. **Importante**: não há uma conta 0 (zero).

Entrada

A entrada é composta por vários casos de teste. Cada caso de testes inicia por uma linha contendo um inteiro N (1<=N<=1000), informando a quantidade de contas daquele caso. Quando N for informado igual a 0 (zero), o programa deve parar imediatamente, sem processar aquele caso. Seguem N linhas contendo, cada uma, as informações de uma conta. Cada linha vai ter o código da conta C (com 14 dígitos), o tipo (o caractere A para analítica ou o caractere S para sintética). Caso a conta seja sintética, terá em seguida dois valores reais (com duas casas decimais): o crédito CR e o débito DE (0,00 <= CR e DE <= 99.999.999,99).

Saída

A saída deve conter N linhas, uma para cada conta, na ordem em que aparecem na entrada. Cada linha deve conter três informações, separadas por um espaço em branco: o código C da conta, retirando os zeros à esquerda e direita; o total de créditos daquela conta; e o total de débitos daquela conta. Imprima uma linha em branco após cada caso de testes na saída. **Leia e imprima na entrada e saída padrão.**

Exemplo de entrada	Respectiva saída
7 00000000000001 A 00000000000011 A 00000000000111 A 00000000001111 S 12.00 13.00 00000000001112 S 4.00 0.00 00000000000112 A 00000000001121 S 11.00 15.00 1 00001230400000 A 0	1 27.00 28.00 11 27.00 28.00 111 16.00 13.00 1111 12.00 13.00 1112 4.00 0.00 112 11.00 15.00 1121 11.00 15.00 12304 0.00 0.00

Autor: Inácio "Capitão", membro da Equipe -, finalista da Maratona de Programação da SBC 2006.

Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema D

Concatenação de Primos

Nome do arquivo fonte: primos.**{c, cpp** ou **java}**

Uma cadeia de caracteres é dita *concPrima* quando é formada apenas por dígitos (0-9), sem zeros à esquerda, e quando é possível formar essa cadeia de caracteres por sub-cadeias concatenadas de tal forma que cada uma dessas sub-cadeias seja um número primo (informalmente, um número primo é divisível apenas por 1 ou ele mesmo, sendo que, por definição, 1 não é um número primo e 2 é um número primo).

Exemplos

"213": é uma cadeia *concPrima,* pois a cadeia pode ser formada pela concatenação de "2" e "13", sendo que 2 e 13 são ambos números primos;
"02": não é *concPrima*, pois começa por um zero à esquerda (atenção a essa regra!);
"a23": não é concPrima, pois é formada por pelo menos um caractere não-dígito. IMPORTANTE: esse caso de testes NÃO pode aparecer na entrada abaixo, pois só podem aparecer dígitos nas cadeias de caracteres informadas;
"4842": não é *concPrima,* pois não há como concatenar números primos para formar essa cadeia de caracteres.

Entrada

A entrada inicia por um inteiro C (C menor que 200) em uma linha isolada informando quantas cadeias de caracteres aparecem na entrada. Cada uma das C linhas seguintes na entrada possui uma cadeia de caracteres formada exclusivamente por no máximo 5 dígitos (0-9) e sem espaços em branco. **Leia da entrada padrão.**

Saída

Para cada cadeia de caracteres na entrada, seu programa deve imprimir em uma linha isolada na saída o texto "sim" (sem aspas) quando a respectiva cadeia de caracteres da entrada for uma cadeia de caracteres *concPrima*, conforme definido pelo enunciado; ou ainda imprimir o texto "nao" (sem aspas e til), quando aquela cadeia de caracteres não for uma *concPrima*. **A saída deve ser impressa na saída padrão.**

Exemplo de entrada	Respectiva saída
8 4842 213 5 02 0 1 13235 41003	nao sim sim nao nao nao sim nao

Autor: Prof. Carlos de Salles.

Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema E

Cafufa Duplo

Nome do arquivo fonte: cafufa.**{c, cpp** ou **java}**

Vocês conhecem o *cafufa*? Cafufa é um jogo muito interessante (principalmente para você aprender a tabuada!) no qual os participantes de maneira ordenada fazem uma contagem simples (1, 2, 3, 4, 5, 6...) entretanto, quando você chega em um múltiplo de 7 ou terminados em 7, em vez de contar o número você fala "Cafufa".

Exemplo de Cafufa com o 7 de 1 a 20.

1, 2, 3, 4, 5, 6, Cafufa, 8, 9, 10, 11, 12, 13, Cafufa, 15, 16, Cafufa, 18, 19, 20.

Entretanto, um grupo de alunos de Ciência da Computação, já estão achando esse jogo sem graça. Eles já jogaram tantas vezes que já não tem mais emoção. Então alguém teve a brilhante ideia: vamos duplicar o cafufa. Agora, em vez de ser múltiplos e terminados em 7, os múltiplos e terminados em dois números quaisquer (determinados previamente) gerarão o termo "Cafufa"! Veja o exemplo para os números 7 e 5:

Exemplo de Cafufa Duplo para 5 e 7 de 1 a 20.

1, 2, 3, 4, Cafufa, 6, Cafufa, 8, 9, Cafufa, 11, 12, 13, Cafufa, Cafufa, 16, Cafufa, 18, 19, Cafufa.

Entrada

Para cada caso da entrada serão informados três números: *N*, *p* e *d*. *N* indica a quantidade de números que entrarão na contagem (1<=N<=1000), *p* e *d* são os números que gerarão a mensagem "Cafufa" segundo as regras do jogo (0 < *p, d* < 10). A entrada termina quando *N*, *p* e *d* forem informados iguais a zero. **Leia da entrada padrão.**

Saída

Para cada cada caso da entrada deve ser informado na saída em uma única linha a contagem no cafufa duplo. Os números devem ser separados por vírgula (,) e espaço em branco e deve haver um ponto (.) ao final da saída. Note que o texto "Cafufa" deve ser impresso com o C maiúsculo. **Imprima na saída padrão.**

Exemplo de entrada	Respectiva saída
20 7 5 5 2 3 0 0 0	1, 2, 3, 4, Cafufa, 6, Cafufa, 8, 9, Cafufa, 11, 12, 13, Cafufa, Cafufa, 16, Cafufa, 18, 19, Cafufa. 1, Cafufa, Cafufa, Cafufa, 5.

Autor: Vanessa Leite, finalista da Maratona de Programação da SBC – Equipe UFMA Diott 2007.

Seletiva Interna da UFMA 2010

Departamento de Informática – 16 de Julho de 2010

Problema F

Partidas de Tênis de Mesa

Nome do arquivo fonte: tenis.**{c, cpp** ou **java}**

Partidas de tênis de mesa são disputadas em sets. Um jogador vence um set quando atinge 11 pontos desde que seu adversário tenha totalizado 9 pontos ou menos. Caso o set não tenha terminado quando um jogador atingir 11 pontos, o set será vencido pelo jogador que conseguir abrir primeiro uma vantagem de 2 pontos de seu adversário. Partidas nacionais são disputadas em melhor de 5 sets (o primeiro que vencer 3 sets ganha a partida) e partidas internacionais são disputadas em melhor de 7 sets (o primeiro que vencer 4 sets ganha a partida).

O chinês Xing-Ling está disputando uma série de partidas de exibição contra o jogador sueco Ovo-Vader. Algumas partidas são disputadas em melhor de 5 sets e outras em melhor de 7 sets. Você anotou cuidadosamente o placar final em sets de todas as partidas. Acontece que só resolveu começar a anotar o placar de cada set depois de um tempo, de memória. Claro: sua memória falhou e esqueceu alguns placares.

Agora precisa da ajuda de seu computador. Sabendo que em todas as partidas, o número de pontos de Xing-Ling foi o **máximo possível** e, como segundo critério, a soma de pontos de Ovo-Vader foi a **mínima possível**, então informe a soma de pontos de Xing-Ling e Ovo-Vader em cada partida, em todos os sets. **IMPORTANTE**: você lembra que nenhum jogador em nenhum set passou da marca de 20 pontos.

Entrada

A entrada inicia por uma linha contendo um inteiro P (P<201) informando a quantidade total de partidas anotadas. A seguir, são informados o placar de P partidas. O placar de uma partida incia por uma linha contendo dois inteiros X (0<=X<=4) e O(0<=O<=4), que são respectivamente o número de sets vencidos naquela partida por Xing-Ling e por Ovo-Vader. A seguir, aparecem X+O linhas com o placar de cada set. O placar de cada set é formado por um par de inteiros A e B, onde A é o número de pontos feito naquele set por Xing-Ling e B é o número de pontos feito naquele set por Ovo-Vader. A e B são inteiros variando de -1 a 20, inclusive. Quando A for informado igual a -1, significa que esqueceu o número de pontos de Xing-Ling naquele set. Quando B for informado igual a -1, significa que esqueceu o número de pontos de Ovo-Vader naquele set. Quando ambos A e B forem iguais a -1 é porque esqueceu os dois placares naquele set. **IMPORTANTE:** não há na entrada nenhum set em que o placar está anotado com totais inválidos ou que não podem se tornar válidos ao se substituir os totais esquecidos (-1) pelo total ocorrido em pontos.

Saída

Para cada partida informada na entrada, seu programa deve imprimir na saída dois inteiros T1 e T2, onde T1 corresponde à soma de pontos de todos os sets de Xing-Ling e mais T2 corresponde à soma de pontos de todos os sets de Ovo-Vader. **Lembre-se**: o total de pontos de T1 deve ser maximizado. Depois disso, o total de pontos T2 deve ser minimizado.

Exemplo de entrada	Respectiva saída
3 3 0 11 5 11 8 11 9 0 3 -1 14 -1 11 14 -1 4 3 11 6 -1 5 8 -1 -1 -1 9 11 14 -1 -1 18	33 22 35 41 93 85

Autor: Prof. Carlos de Salles.