MARATONA DE PROGRAMAÇÃO

InterFatecs

Americana - 2018

Final 25/08/2018

Caderno de Problemas

Patrocínio:

FUNDAÇÃO FAT

IBM

KEYRUS
insight into value

CENGAGE

Realização:

Fatec Americana

CPS Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

www.interfatecs.com.br

1 Instruções

Este caderno contém 10 problemas – identificados por letras de A até J, com páginas numeradas de 3 até 24. Verifique se seu caderno está completo.

Informações gerais

1. Sobre a competição
   (a) A competição possui duração de 5 horas (início as 13:00h término as 18:00h);
   (b) NÃO é permitido acesso a conteúdo da Internet ou qualquer outro meio eletrônico digital;
   (c) É permitido somente acesso a conteúdo impresso (cadernos, apostilas, livros);
   (d) É vedada a comunicação entre as equipes durante a competição, bem como a troca de material de consulta entre elas;
   (e) Cada equipe terá acesso a 1 computador dotado do ambiente de submissão de programas (BOCA), dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;
   (f) NÃO é permitido o uso de notebooks ou outro tipo de computador ou assistente pessoal;
   (g) Os problemas têm o mesmo valor na correção.
2. Sobre o arquivo de solução e submissão:
   (a) O arquivo de solução (o programa fonte) deve ter o mesmo nome que o especificado no enunciado (logo após o título do problema);
   (b) confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução.
   (c) NÃO insira acentos no arquivo-fonte.
3. Sobre a entrada
   (a) A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
   (b) Seu programa será testado em vários casos de teste válidos além daqueles apresentados nos exemplos. Considere que seu programa será executado uma vez para cada caso de teste.
4. Sobre a saída
   (a) A saída do seu programa deve ser escrita na saída padrão;
   (b) Não exiba qualquer outra mensagem além do especificado no enunciado.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem A

InterFatecs Secrets

Source file: secrets.{ c | cpp | java | py }
Author: Lucio Nunes de Lira (Fatec São Paulo)

Silvio is a storyteller whose stories are sometimes true and other times not. Recently, Silvio told his friends of the Interfatecs Secrets legend. The legend states that, after the InterFatecs Contest final phase, the winner team, its coach and the contestants family members are invited to take part in a closed event, whose activities are as secret as the Silvio information sources.

According to Silvio, there are three different access levels to activities during the event. The highest is named *ultimate*, the intermediate is named *premium* and the lowest is named *basic*. In order to distinguish the guests' access level, each one receives a code, not necessarily unique, that must be presented at the event entrance. After receiving the code, the doorman types it into a *software* that generates a natural number indicating the guest access level.

The algorithm used by the software to obtain the natural number considers the code symbols and their positions. The symbols that may be used in a code are lowercase letters $(a - z)$ and digits $(0 - 9)$. Every symbol has a numeric value. For lowercase letters, the value is the letter position in the alphabet $(a = 1, b = 2, z = 26)$. For digits, the value is the numeric value of the digit. The code total value is the sum of the value of each symbol multiplied by its position.

To get access to the *ultimate* level, the generated number must be the result of the factorial of some natural number n $(n! = n * (n-1) * (n-2)...(n-n)$, by definition $0! = 1)$; to get access to the *premium* level, the number must belong to the Fibonacci Sequence (the first and second terms are 1, and any other is the sum of the two immediately preceding terms); the *basic* level requires the number to be prime (prime numbers have only two natural divisors, 1 and itself). If the number is not in the previous categories, the code is invalid.

Whether it is a true story or not, your goal is to create a program that does the same as what was aforementioned. Note that the program should always indicate the highest possible level for a code.

Input

The input has many lines, each one with a code C $(1 \leqslant length(C) \leqslant 276)$.

Output

For each input line, print the number that was generated for the code followed by a colon, a space and, finally, the code access level ($ultimate$, $premium$ or $basic$). If the code is invalid, print an exclamation '!' as its access level (without quotes). Each output line ends with a line break.

Example of Input 1	Example of Output 1
abc 123 100 001 700	14: ! 14: ! 1: ultimate 3: premium 7: basic

Example of Input 2	Example of Output 2
cba 321 a2 z	10: ! 10: ! 5: premium 26: !

Example of Input 3	Example of Output 3
zzzzzzzzzz aaaaaaaaaa 99999 00000	1430: ! 55: premium 135: ! 0: !

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema B

Fliperama

Arquivo fonte: fliperama.{ c | cpp | java | py }

Autor: Silvio do Lago Pereira e Lucio Nunes de Lira (Fatec São Paulo)

Fliperamas são muito divertidos! Quem presenciou o auge desses jogos sabe o quão emocionante era competir com os amigos buscando a maior pontuação da máquina. Existem muitos tipos de fliperamas, com os mais variados funcionamentos e objetivos. Aqui, estamos interessados em um tipo específico, ilustrado simplificadamente na Figura 1, que funciona da seguinte maneira:

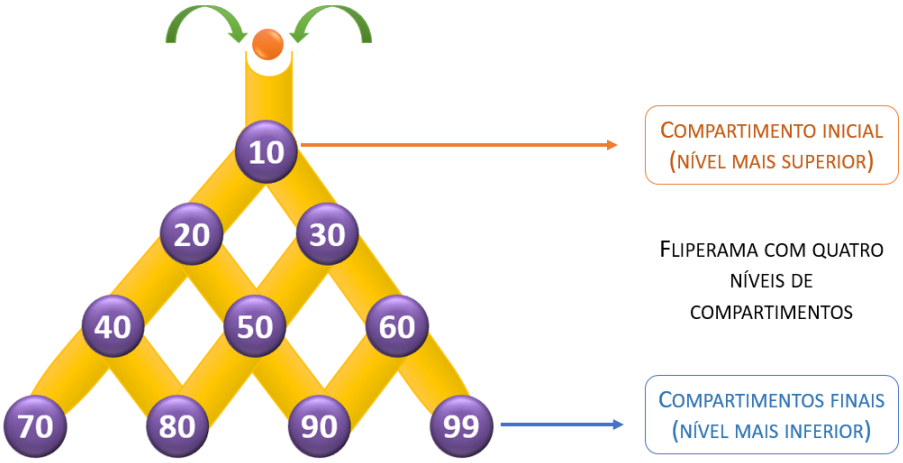
a) O jogador insere uma esfera no topo do fliperama, que tem diversos compartimentos com pontos;

b) A esfera cai, acumulando os pontos do compartimento inicial;

c) A queda continua pelas bifurcações e a esfera atinge compartimentos em níveis inferiores, acumulando a pontuação daqueles em que passa;

d) Após acumular os pontos do compartimento no nível mais inferior, o jogo termina e a pontuação total é exibida.

Figura B.1: Possível configuração do fliperama.



Note que todo compartimento, exceto os do nível mais inferior, tem uma bifurcação que dá acesso ao compartimento inferior esquerdo e inferior direito, sendo impossível que a esfera passe pelos dois em um único jogo e acesse mais de um compartimento no mesmo nível.

Podemos representar esse tipo de fliperama com valores numéricos dispostos em linhas e colunas. Nesta representação um compartimento em um nível inferior tem como conexões no nível superior àquele da linha imediatamente acima (se houver) e da coluna imediatamente à esquerda (se houver). Vide *Exemplo de Entrada 1*, que corresponde ao fliperama da Figura 1.

Você, excelente em programação, deve construir um programa que receba uma representação em linhas e colunas de um desses fliperamas e exiba o máximo e o mínimo de pontos que um jogador pode atingir.

Entrada

A entrada é composta pela quantidade N de níveis do fliperama $(1 \leqslant N \leqslant 500)$; por N linhas com C compartimentos $(C = N - numeroDaLinha + 1)$ com P pontos cada $(-2^{16} \leqslant P \leqslant 2^{16})$.

Saída

A palavra '$maximo$' (minúscula, sem acentuação e sem aspas), seguida por ':' (sem aspas), um espaço, o máximo de pontos possível e uma quebra de linha; a palavra '$minimo$' (minúscula, sem acentuação e sem aspas), seguida por ':' (sem aspas), um espaço, o mínimo de pontos possível e uma quebra de linha.

Exemplo de Entrada 1

```
4
10 30 60 99
20 50 90
40 80
70
```

Exemplo de Saída 1

```
maximo: 199
minimo: 140
```

Exemplo de Entrada 2

```
4
23 32 55 73
27 50 22
14 60
69
```

Exemplo de Saída 2

```
maximo: 183
minimo: 122
```

Exemplo de Entrada 3

```
3
4 3 9
1 5
4
```

Exemplo de Saída 3

```
maximo: 16
minimo: 9
```

Exemplo de Entrada 4

```
3
4 1 9
3 5
4
```

Exemplo de Saída 4

```
maximo: 14
minimo: 10
```

Exemplo de Entrada 5

```
2
-5 0
-2
```

Exemplo de Saída 5

```
maximo: -5
minimo: -7
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

Semáforo

Arquivo fonte: semaforo.{ c | cpp | java | py }
Autor: Julio Lieira (Fatec Lins)

Um cruzamento entre duas avenidas mais conhecidas da cidade de Fatecsburgo é controlado por quatro semáforos. Lá cada um é conhecido por uma letra (A, B, C e D). Cada semáforo possui um tempo em segundos em que fica aberto (no verde) e um tempo em amarelo. O tempo em verde pode ser diferente para cada semáforo e o tempo em amarelo (transição entre verde e vermelho) é sempre o mesmo para todos os semáforos. Somente um semáforo é aberto por vez e sempre na mesma sequência (primeiro o semáforo A, depois o B, em seguida o C e por último o D, voltando ao A, o B e assim sucessivamente). Nesse cruzamento em específico não se perde tempo na transição entre o amarelo e o vermelho, pois neste mesmo instante, abre-se o semáforo na sequência.

Supondo que nessa cidade os semáforos trabalham sem interrupção, nem com chuva brava eles param, e que no primeiro segundo o semáforo A inicia em Verde, deseja-se saber qual semáforo estará aberto em um determinado instante do tempo.

Entrada

A primeira linha da entrada possui cinco valores inteiros A, B, C, D, Y ($1 \leqslant A, B, C, D, Y \leqslant 60$), separados por um espaço, representando o tempo em segundos para o sinal Verde dos semáforos A, B, C e D. O valor Y corresponde ao tempo em segundos para o sinal Amarelo que é igual para os quatro semáforos. A segunda linha contém três valores inteiros H, M e S ($0 \leqslant H \leqslant 55\text{x}10^4, 0 \leqslant M, S \leqslant 59$), separados por um espaço, representando o tempo em que se deseja saber qual semáforo está aberto, no formato H=horas, M=minutos e S=segundos. Considere que pelo menos um dos três tempos sempre será maior que zero.

Saída

Imprima na saída a letra do semáforo que está aberto no instante de tempo desejado, seguida da palavra VERDE. Caso no instante de tempo desejado nenhum semáforo esteja aberto, imprima a letra seguida da palavra AMARELO do semáforo que esteja no sinal Amarelo. A saída deve terminar com uma quebra de linha.

Exemplo de Entrada 1	**Exemplo de Saída 1**
6 4 5 3 2 0 0 1	A VERDE

Exemplo de Entrada 2	**Exemplo de Saída 2**
6 4 5 3 2 0 0 6	A VERDE

Exemplo de Entrada 3	Exemplo de Saída 3
6 4 5 3 2 0 0 7	A AMAREO

Exemplo de Entrada 4	Exemplo de Saída 4
6 4 5 3 2 0 0 23	D VERDE

Exemplo de Entrada 5	Exemplo de Saída 5
6 4 5 3 2 0 0 59	A AMARELO

Exemplo de Entrada 6	Exemplo de Saída 6
6 4 5 3 2 0 2 7	D VERDE

Exemplo de Entrada 7	Exemplo de Saída 7
6 4 5 3 2 1 2 7	B VERDE

Exemplo de Entrada 8	Exemplo de Saída 8
20 15 20 12 2 0 0 45	C VERDE

Exemplo de Entrada 9	Exemplo de Saída 9
20 15 20 12 2 0 1 32	A VERDE

Exemplo de Entrada 10	Exemplo de Saída 10
20 15 20 12 2 550000 51 1	C AMARELO

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

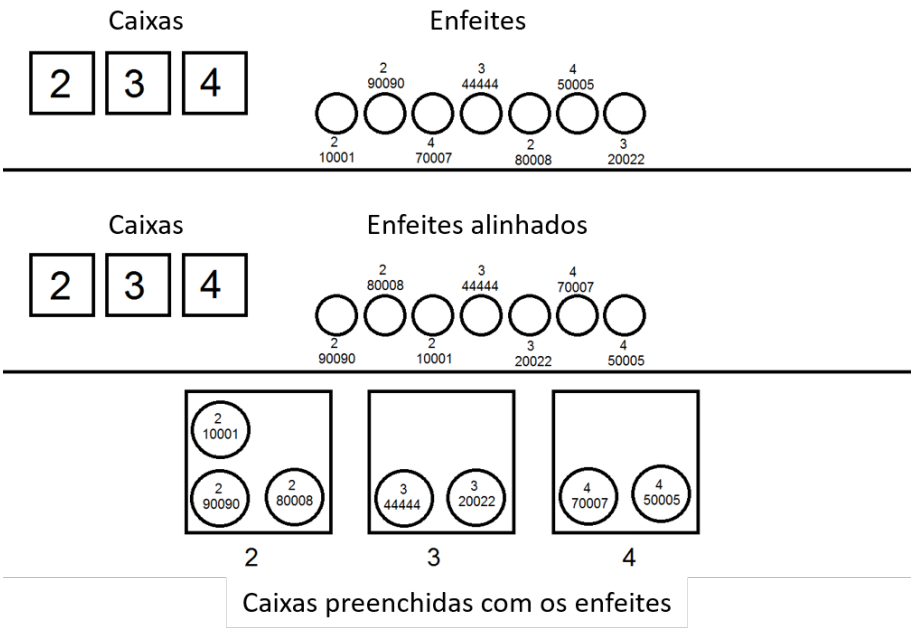
Enfeites de Natal

Arquivo fonte: enfeites.{ c | cpp | java | py }
Autor: Anderson V. Araujo (UFMS)

Com o término da época natalina, o shopping center da cidade tem que guardar todas as decorações de Natal. Como os funcionários responsáveis pela montagem e desmontagem das decorações são temporários, o shopping possui um sistema de rotulagem para que os enfeites sejam sempre guardados no mesmo lugar e da mesma forma. Cada enfeite possui uma etiqueta com o número da caixa onde deve ser guardado e um código. Dentro de cada caixa, os enfeites devem ser colocados em ordem decrescente de código, para que os enfeites com o código maior fiquem mais no fundo. No próximo ano as caixas deverão ser abertas em ordem crescente de seus números.

Para agilizar o trabalho, os funcionários desejam enfileirar os enfeites antes de guardá-los, na ordem em que eles devem ser guardados, como mostrado na Figura D.1.

Figura D.1: Exemplo de organização dos enfeites nas respectivas caixas.



Sua tarefa é ajudar os funcionários a organizar os enfeites para que fique mais fácil a organização da decoração de natal do ano seguinte.

Entrada

A entrada começa com um número **N** (0 < N < 10000) que indica o número de enfeites a serem guardados, seguido de **N** linhas contendo dois inteiros positivos **C** ($0 \leq C < 10000$) e **D** ($10000 \leq D < 2^{31} - 1$), onde **C** representa o número da caixa e **D** o código de cada enfeite.

Saída

A saída deve conter **N** linhas, cada uma com o código dos enfeites na ordem em que eles devem ser enfileirados para serem armazenados em suas respectivas caixas. Finalize com uma quebra de linha.

Exemplo de Entrada 1

```
7
2 10001
2 90090
4 70007
3 44444
2 80008
4 50005
3 20022
```

Exemplo de Saída 1

```
90090
80008
10001
44444
20022
70007
50005
```

Exemplo de Entrada 2

```
10
4 98622
9 91
9 911
5 387
9 376
10 3
1 1505
4 14
6 102
9 105
```

Exemplo de Saída 2

```
1505
98622
14
387
102
911
376
105
91
3
```

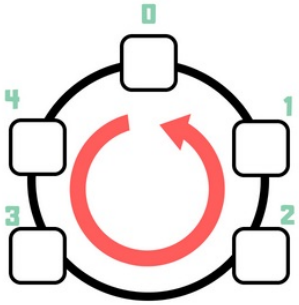
MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

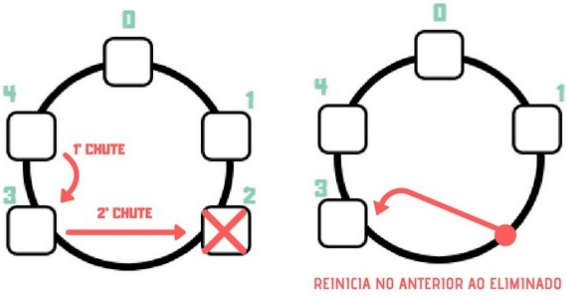
Bola de Papel

Arquivo fonte: papel.{ c | cpp | java | py }
Autor: Érico de Souza Veriscimo (ETEC de Guaianazes)

Uma turma do Ensino Médio Integrado de uma determinada ETEC gosta muito de futebol e em seus intervalos fazem uma bola de papel para jogarem nos corredores da instituição. Porém, um dos alunos teve uma ideia para ficar ainda mais divertida tal brincadeira. Cada bola de papel feita com uma folha dura apenas 2 chutes, ou seja, a cada camada de folha adicionada à bola, a mesma durará mais 2 chutes. Os alunos ficaram em disposição circular como mostra a imagem e serão classificados de forma horária por um número de 0 a N-1 e o aluno que estiver com a bola chutará para o seu amigo a direita, ou seja, em sentido anti-horário, como mostra a figura a baixo.



O jogo acontecerá da seguinte maneira: será eleito um dos alunos para começar com a bola, e serão mencionadas quantas folhas foram utilizadas para a confecção da bola. O aluno que começar com a bola irá chutar a bola para o seu amigo da direta e o mesmo também chutará para o seu amigo da direita, isso se repetirá até que a bola não suporte mais chutes. Assim, o aluno em que parar a bola será eliminado da brincadeira e confeccionará uma nova bola, porém com 1 folha a mais. Isso se repetirá até que apenas um aluno fique na brincadeira e este aluno será o vencedor. Abaixo um exemplo no qual a bola inicia com apenas 1 folha e o aluno que começará com a bola é o aluno número 4. O vencedor será o aluno de número 1.



Entrada

A entrada consiste de um único caso de teste, composto por 1 linha com três inteiros Q $(1 \leqslant Q \leqslant 200)$, F $(1 \leqslant F \leqslant 500)$ e A $(1 \leqslant A \leqslant 200)$ respectivamente a quantidade de alunos, a quantidade de folhas que serão utilizadas para confeccionar a primeira bola e o aluno que começara com a bola.

Saída

Para cada caso de teste imprima o número do aluno vencedor. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
5 1 4	1

Exemplo de Entrada 2	Exemplo de Saída 2
2 5 1	0

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema F

Fiscal de vestibular

Arquivo fonte: fiscal.{ c | cpp | java | py }
Autor: Lucio Nunes de Lira (Fatec São Paulo)

Todos já passamos por processos seletivos de faculdades, os conhecidos vestibulares, porém, poucos experimentaram o papel de fiscal. Uma das atribuições de um fiscal é manter em ordem os documentos manipulados durante o vestibular. Inicialmente, a coordenação entrega os documentos ao fiscal, em seguida ele entrega aos candidatos e, ao final, recebe dos candidatos e os entrega à coordenação.

Cada vestibular tem suas particularidades, aquele em que estamos interessados tem a característica do duplo turno, que consiste em um intervalo entre as partes do processo seletivo. No primeiro turno é aplicada a prova objetiva, no segundo a redação. Além disso, todo candidato possui uma folha de identificação, que deve ser assinada depois de cada turno, confirmando sua presença.

Há candidatos que se ausentam em algum dos turnos, ou até em ambos. A coordenação exige que o fiscal devolva as folhas de identificação em ordem alfabética, porém com os candidatos que participaram de ambos os turnos na frente, os que participaram apenas da prova objetiva em segundo, os presentes somente na redação em terceiro e, por último, os que não participaram de nenhum turno.

Você foi contratado para fazer um programa que ajude a coordenação na conferência dos documentos devolvidos pelos fiscais. O programa lerá diversos registros, cada um composto por um número com a posição em que o fiscal colocou a folha de identificação, uma letra com a situação de presença e o nome do respectivo candidato. Ao final, o programa deverá comparar os registros com a ordem exigida pela coordenação e informar quantos estão em posição **incorreta**.

Entrada

A entrada é composta pela quantidade Q de candidatos $(1 \leqslant Q \leqslant 10^5)$; por Q linhas com a posição P da folha de identificação; a letra da situação S de presença do candidato ('*A*', '*O*', '*R*' ou '*N*' simbolizando *ambas*, *objetiva*, *redação* e *nenhuma*, respectivamente); pelo nome N $(1 \leqslant tamanho(N) \leqslant 50)$, minúsculo, não acentuado e único entre os candidatos. Os itens são separados pelo caractere '*;*'.

Saída

A quantidade de folhas de identificação em posição **incorreta**. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
5 3;O;victoria 2;A;lucio 4;R;maria antonia 5;A;airton 1;R;luciene	3

Exemplo de Entrada 2

```
5
1;N;shane
2;N;rick
3;N;morgan
4;N;carl
5;N;andrea
```

Exemplo de Saída 2

```
4
```

Exemplo de Entrada 3

```
4
4;R;mariacecilia
2;R;maria cecilia
3;R;mariaceci
1;R;maria ceci
```

Exemplo de Saída 3

```
0
```

Exemplo de Entrada 4

```
3
2;A;jonas atilo
1;A;jonas bruno
3;A;jonas claudio
```

Exemplo de Saída 4

```
2
```

Exemplo de Entrada 5

```
3
2;O;jonas atilo
3;N;jonas bruno
1;A;jonas claudio
```

Exemplo de Saída 5

```
0
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

Jogo das Palavras

Arquivo fonte: palavras.{ c | cpp | java | py }
Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

João e Maria são dois irmãos gêmeos que passam mais tempo do que gostariam juntos. E então, brigam. Brigam muito. Seus pais se incomodam com isso, mas não podem evitar que os dois pirralhos fiquem juntos. Como também não estão indo muito bem na escola, principalmente na disciplina de Língua Portuguesa, decidiram propor atividades que pudessem fazer em dupla, de maneira que brigassem menos e aprendessem mais. Uma dessas atividades é um joguinho muito simples: um deles fala uma palavra e o outro precisa responder uma outra palavra que seja uma permutação da primeira. Por exemplo, se João começa e fala "Gato", Maria pode responder "Toga", que é uma palavra que tem exatamente as mesmas letras de "Gato", apenas em ordem diferente. Outro exemplo poderia ser um dos dois falar "Rolha" e o outro responder "Olhar", viu como é fácil? Para melhorar a qualidade da atividade, o pai das crianças gostaria que fosse criado um programa tira-teima, para decidir se, para duas palavras informadas, uma é permutação da outra. Essa é a sua tarefa neste problema.

Entrada

A entrada é composta por um único caso de teste, expresso em uma linha contendo duas palavras com até 30 caracteres de comprimento cada uma. Você poderá assumir que apenas caracteres alfabéticos são usados na grafia das palavras.

Saída

O programa deve imprimir uma letra maiúscula "S" se a segunda palavra é uma permutação da primeira e "N" em caso contrário. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
`Gato Toga`	`S`

Exemplo de Entrada 2	Exemplo de Saída 2
`Olhar Rolha`	`S`

Exemplo de Entrada 3	Exemplo de Saída 3
`Banana Batata`	`N`

Exemplo de Entrada 4	Exemplo de Saída 4
`Saco Socas`	`N`

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

Layout Linear

Arquivo fonte: layoutlinear.{ c | cpp | java | py }
Autor: Leandro Luque (Fatec Mogi das Cruzes)

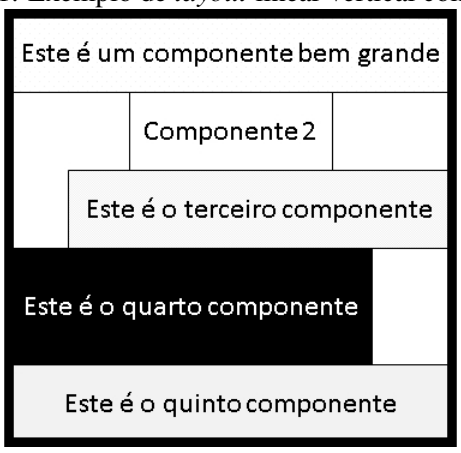
Eduardo, um programador com deficiência visual, está um pouco chateado com o W3C, consórcio responsável, entre outras coisas, pela especificação da linguagem HTML. Ele gostaria de ter recursos de gerenciamento de *layout* na própria linguagem HTML ou em alguma linguagem relacionada, como o CSS. Afinal, ele não consegue atualmente desenhar interfaces adequadas, por não enxergar. Com gerenciadores de *layout*, ele poderia especificar o conteúdo que deseja desenhar e o gerenciador cuidaria da organização gráfica do conteúdo.

No Android, por exemplo, existem vários gerenciadores de *layout*, como o linear (que organiza o conteúdo na horizontal ou vertical) e o relativo (que organiza o conteúdo no que diz respeito à relação existente entre os componentes de interface).

Mônica, sua namorada, decidiu ajudá-lo. Ela iniciou a implementação de um gerenciador de *layout* simples que pode auxiliar Eduardo na construção de interfaces gráficas. Por ter alguma experiência com Android, escolheu o *layout* linear. Ainda, por ser sua primeira iniciativa desta natureza, ela optou por implementar apenas o *layout* linear de orientação vertical.

Um *layout* linear vertical organiza verticalmente os componentes de interface nele inseridos. Em outras palavras, um componente por linha. Veja, no exemplo a seguir, um *layout* linear com 5 rótulos.

Figura H.1: Exemplo de *layout* linear vertical com 5 rótulos



Cada componente no *layout* de Mônica possui 3 atributos: altura, largura e alinhamento horizontal. A altura representa a quantidade de pixels que o componente ocupa na vertical. A largura pode ser especificada de 2 formas diferentes:

- Por meio da quantidade de pixels que o componente ocupa na horizontal (podendo inclusive esconder o seu conteúdo) OU
- Por meio de uma política de largura, que pode ser $WRAPCONTENT$ ou $MATCHPARENT$. Um componente com política $WRAPCONTENT$ ocupa apenas o espaço necessário para exibir todo o seu conteúdo, podem sobrar espaços à esquerda e/ou à direita dele. Como Mônica está permi-

tindo apenas que rótulos (textos) sejam inseridos no gerenciador de *layout*, o espaço necessário para exibir o conteúdo de um texto é equivalente ao tamanho do texto multiplicado por uma constante, que representa a largura dos caracteres na fonte monoespaçada utilizada. Um componente com política $MATCHPARENT$, por outro lado, tem largura sempre igual à do gerenciador de *layout* no qual está inserido, não podendo, assim como no caso do $WRAPCONTENT$, ter seu conteúdo escondido. Em outras palavras, se a largura atual do gerenciador de *layout* for menor que a largura mínima para exibir um componente, o gerenciador de layout terá sua largura aumentada.

O alinhamento horizontal é usado quando um componente não ocupa toda a largura do gerenciador de *layout* no qual está inserido. Ele pode assumir três valores: E (à esquerda), C (Centralizado) ou D (à direita).

No exemplo apresentado na Figura H.1, o componente 2 tem alinhamento C, o terceiro componente D e o quarto E.

Ajude Mônica a testar o código que está desenvolvendo e implemente o *layout* linear vertical especificado por ela. Dado o conteúdo e os atributos que definem os componentes a serem inseridos em um *layout* vertical, escreva um programa que determina a altura total do *layout* e a quantidade de pixels à esquerda e à direita de cada um de seus componentes.

Entrada

A primeira linha da entrada contém um número inteiro N ($0 < N \leqslant 100$), contendo o número de componentes que serão inseridos no *layout* linear vertical. As próximas N linhas contêm cinco valores separados por um espaço em branco. O primeiro é um número inteiro A ($0 < A \leqslant 500$), representando a altura do componente em pixels. O segundo valor é um caractere F ($F \in ['A', 'P']$), sem aspas, indicando se a largura do componente será especificada por meio de um valor absoluto (A) ou por meio de uma política de largura (P). Caso F seja igual a 'A', o próximo valor é um número inteiro L ($0 < L \leqslant 500$) indicando a largura do componente. Caso F seja igual a 'P', o próximo valor é uma das seguintes sequências de caracteres: $WRAPCONTENT$ ou $MATCHPARENT$. O quarto valor é um caractere Z ($Z \in ['E', 'C', 'D']$), sem aspas, representando o alinhamento horizontal do componente: E-Esquerda, C-Centro, D-Direita. O quinto e último valor de cada linha é uma cadeia de caracteres C, com tamanho máximo 100, representando o conteúdo do componente. A última linha do caso de teste contém um número inteiro T ($0 < T \leqslant 20$), indicando o tamanho horizontal em pixels que cada caractere do componente ocupa.

Saída

Como saída, o seu programa deverá imprimir $N + 1$ linhas de resultado. A primeira linha deve exibir a altura total do *layout*. As próximas N linhas devem imprimir dois números inteiros T e Q, separados por um espaço em branco, representando o tamanho do espaço (em pixels) à esquerda e à direita de cada componente respectivamente. Caso o tamanho do espaço seja um número real, utilize o truncamento para exibir o resultado. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
5 10 P MATCHPARENT C Este é um componente bem grande 20 P WRAPCONTENT C Componente2 20 A 30 D Este é o terceiro componente 10 P WRAPCONTENT E Este é o quarto componente 20 P MATCHPARENT C Este é o quinto componente 1	80 0 0 10 10 1 0 0 5 0 0

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

Procura-se

Arquivo fonte: emprego.{ c | cpp | java | py }
Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Catarina precisa contratar um novo funcionário para o escritório da empresa. Com medo de repetir insucessos anteriores, pretende caprichar desta vez e contratar alguém com bom perfil e em curto espaço de tempo. O problema que ela enxerga é que muitas vezes entrevista pessoas demais, comprometendo o fator tempo para a contratação, outras vezes entrevista poucas pessoas e escolhe logo, sacrificando nesse caso o aspecto diversidade, pois seria possível que um ótimo candidato surgisse, caso o processo se estendesse um pouco mais. Pesquisando na literatura da área, descobriu um método que despertou muito a sua atenção, chamado Regra dos 37, aplicado para o chamado "problema da secretária". Esse problema envolvia decidir quantas candidatas se deveria entrevistar até que se decidisse contratar alguém para o cargo, assumindo que não existirão duas candidatas com o mesmo resultado na avaliação (ou seja, não há empate) e que, ao final da entrevista, ou se contrata a candidata (e ela sempre vai aceitar a oferta) ou a dispensa (e nesse caso ela nunca mais vai aceitar outra proposta sua). A Regra dos 37 prescreve que após termos conhecido as 37% primeiras candidatas, devemos continuar e escolher a primeira que surgir que tenha avaliação melhor que as anteriores, encerrando ali o processo. No caso infeliz em que a melhor secretária esteja justamente no pequeno grupo inicial de 37%, devemos seguir até o fim das candidatas e ficar com a última. Existem alguns casos especiais: se existe apenas uma candidata, escolhemos essa única (que é também a primeira colocada). Se temos duas, faz sentido escolher sempre a segunda. No caso de termos 3, devemos entrevistar a primeira e em seguida entrevistar a segunda: caso ela seja melhor que a primeira, será escolhida, senão, teremos que escolher a terceira. No caso de termos 4 candidatas, entrevistamos duas e, se a terceira for melhor, ficamos com ela, senão contratamos a última. Os casos restantes passam a funcionar pela norma: entrevistamos sem contratar até termos conhecido as 37% primeiras, e a partir daí contratamos a primeira melhor que surgir, e caso nenhuma seja melhor, ficamos com a última. Nossa heroína Catarina se interessou um pouco, mas está receosa, e pretende fazer algumas simulações para avaliar se vale a pena utilizar esse método. Sua ideia é criar várias situações com diversos candidatos (ela não tem a restrição de escolher obrigatoriamente mulheres) com as suas respectivas posições no ranking geral de entrevistados. Um programa de computador determinaria então, para aquele caso, qual seria a pessoa escolhida pela vaga, segundo o método. Após várias simulações produzidas com auxílio desse programa, Catarina espera ter condições de decidir se adotar esse método seria ou não interessante. Como você já deve estar pensando, sua tarefa é construir esse programa que implementa a Regra dos 37 para o tal "problema da secretária".

Entrada

A entrada consiste de um único caso de teste, composto por diversas linhas, cada uma contendo uma string de até 25 caracteres indicando o nome do candidato(a), seguida de um inteiro R informando qual a sua nota na avaliação $(0 \leqslant R \leqslant 1000)$. Quanto maior o valor de R melhor o desempenho do candidato. O conjunto de dados de entrada é finalizado por EOF. O valor de R não se repete entre os candidatos de um mesmo processo seletivo. O nome do candidato corresponde sempre a uma palavra, composta apenas por caracteres alfabéticos. Você pode assumir que sempre haverá pelo menos um e não mais do que 100 candidatos à vaga. Observe os exemplos a seguir, que ilustram tanto casos gerais como especiais.

Saída

Imprima o nome do candidato escolhido pela Regra dos 37, seguido por um espaço em branco e o seu valor R lido da entrada. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
Clara 12 Silvio 35 Antonio 41 Bernadete 57 Julia 23	Antonio 41

Exemplo de Entrada 2	Exemplo de Saída 2
Clara 77 Bernadete 18 Silvio 43 Antonio 44 Julia 22	Julia 22

Exemplo de Entrada 3	Exemplo de Saída 3
Joao 2 Renata 3 Ricardo 4 Paula 1	Ricardo 4

Exemplo de Entrada 4	Exemplo de Saída 4
Maria 37 Jose 23 Pedro 12	Pedro 12

Exemplo de Entrada 5	Exemplo de Saída 5
Paulo 2 Ana 19	Ana 19

Exemplo de Entrada 6	Exemplo de Saída 6
Silvana 40	Silvana 40

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

Transferências de cães

Arquivo fonte: transferencias.{ c | cpp | java | py }
Autor: Lucio Nunes de Lira (Fatec São Paulo)

Um canil possui três enormes casas para seus cães (*Amarela*, *Branca* e *Cinza*) e está com um problema tão grande quanto. Mariazinha, a dona do canil, precisa transferir todos os cães da casa mais velha, a *Amarela*, para a mais nova, a *Cinza*.

Neste canil, os cães são identificados internamente por seus tempos de vida em dias, e não há cães que nasceram na mesma data. Por serem animais territorialistas, existem regras para que o procedimento de transferência seja cumprido adequadamente, evitando problemas de convivência entre os animaizinhos:

a) Uma transferência implica na saída de um cão de uma casa e entrada imediata em outra;

b) Somente um cão pode ser transferido por vez e de uma única casa por vez;

c) Para transferir um cão, a prioridade é do mais jovem da casa;

d) Antes que um cão seja transferido, todos os cães mais jovens da casa destino devem ser transferidos.

Como a casa *Branca* está em desuso, pode ser utilizada para apoio no procedimento de transferência, abrigando temporariamente os cães até que sejam postos no local definitivo, ou seja, pode haver mais de uma transferência por cãozinho, independente da origem e destino.

Mariazinha solicitou gentilmente a sua ajuda para construir um programa que receba as idades em dias de todos os cães e informe qual a quantidade mínima de transferências entre casas para cumprir o procedimento.

Entrada

É composta pela quantidade N de cães $(1 \leqslant N \leqslant 60)$ e por N linhas com I idades distintas $(1 \leqslant I \leqslant 7300)$.

Saída

A quantidade mínima de transferências entre as casas e uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
1 100	1

Exemplo de Entrada 2	Exemplo de Saída 2
2 59 60	3

Exemplo de Entrada 3	Exemplo de Saída 3
5 67 54 2000 1 4	31

Exemplo de Entrada 4	Exemplo de Saída 4
8 1 2 5 7300 7299 7100 10 20	255