ToPAS'19
Torneio de Programação para Alunos do Secundário

Departamento de Informática

http://eventos.fct.unl.pt/topas-lx

Conjunto de Problemas

Faculdade de Ciências e Tecnologia da NOVA

3 de abril de 2019

Este conjunto de problemas deverá conter sete (7) problemas e dezassete (17) páginas.
Se faltar algum problema, por favor avise a organização.

Edição realizada em colaboração com:

Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto

Departamento de Engenharia Eletrónica e Informática, Faculdade de Ciências e Tecnologia, Universidade do Algarve

ToPAS'19
Torneio de Programação para Alunos do Secundário

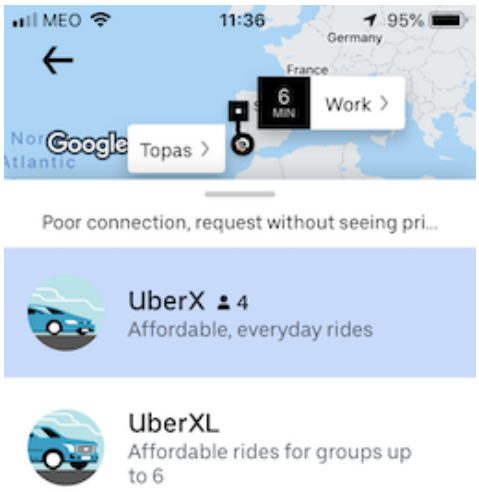
Departamento de Informática – FCT NOVA
3 de abril de 2019

Conteúdo

Problema A: Uber 3
Problema B: Segredos mal guardados 5
Problema C: CSI Portugal 7
Problema D: Tio Patinhas, o avarento? 9
Problema E: Magia com cartas 11
Problema F: Joaquim Moedas 13
Problema G: Ainda os apanhamos 15

Uber

A Uber já serve todas as cidades que têm ToPAS: Lisboa, Porto e Faro. Nessas cidades, há dois tipos de serviço: o UberX, mais barato, e o UberXL, com carros maiores, que levam mais passageiros, mais caro. Para cada serviço estão fixados o custo por minuto e o custo por quilómetro. Em ambos os serviços, o preço obtém-se somando a componente do tempo, a componente da distância e a tarifa base. Complementarmente, existe uma taxa mínima, que será usada caso o preço resultante da soma daquelas três componentes (tempo, distância e tarifa base) seja menor que a taxa mínima. O preço a pagar é calculado no fim da viagem, pelos computadores da Uber.



Ainda assim, queremos ter um programa nosso para fazer também esses cálculos, para verificar que o valor que nos é cobrado está correto.

Tarefa

Escreva um programa que, dados o tipo do serviço, o tempo gasto e a distância percorrida, calcule o preço da viagem.

O tipo do serviço UberX é representado pelo número 1 e o tipo de serviço UberXL é representado pelo número 2; o tempo gasto é um número inteiro não negativo, representando a duração da viagem em minutos; a distância percorrida é um número inteiro não negativo, representando o número de quilómetros do percurso. Para o UberX, o custo por minuto é 10 cêntimos, o custo por quilómetro é 80 cêntimos, a tarifa base é €1.00 e a taxa mínima é €2.50; para o UberXL os valores são 15 cêntimos, €1.20, €1.50 e €3.50, respetivamente.

O preço da viagem deve ser expresso em cêntimos, por um número inteiro.

Input

O input são três números inteiros, dados todos na mesma linha, separados cada um do seguinte por um único espaço. O primeiro número representa o tipo de serviço, o segundo (T) representa a duração da viagem em minutos e o terceiro (D) representa o comprimento do percurso em quilómetros.

Restrições

$0 \leq T \leq 10\,000$ Tempo gasto

$0 \leq D \leq 5\,000$ Distância percorrida

Output

O output será uma única linha na qual figurará o número inteiro que representa o preço da viagem.

Exemplo 1

Input

```
1 20 10
```

Output

```
1100
```

Exemplo 2

Input

```
1 3 1
```

Output

```
250
```

Exemplo 3

Input

```
2 30 40
```

Output

```
5400
```

Segredos mal guardados

O Gil, que é uma criança endiabrada, está numa fase de bisbilhotice e, sempre que pode, vai vasculhar as coisas dos irmãos mais velhos. A Maria, que não quer que o irmão descubra os seus segredos, guarda-os na sua cómoda, cujas gavetas se podem fechar à chave. Mas ainda não percebeu que o irmão consegue aceder ao conteúdo de uma gaveta fechada à chave, retirando a gaveta imediatamente acima, se esta não estiver fechada à chave.

A cómoda tem uma sequência de gavetas iguais, na vertical, e cada gaveta tem uma fechadura independente. Para o Gil, uma gaveta só está *inacessível* se:

- ou é a gaveta de cima da cómoda e está fechada à chave;
- ou é uma das outras gavetas, está fechada à chave e a gaveta imediatamente acima também está fechada à chave.

Dada uma configuração da cómoda (ou seja, o estado das suas fechaduras), pretende-se saber quantas gavetas estão inacessíveis para o Gil.

A figura seguinte apresenta oito configurações de uma cómoda com 4 gavetas, onde "`#`" indica que a gaveta está fechada à chave e "`o`" indica que a gaveta não está fechada à chave. Por baixo de cada configuração, indica-se o número de gavetas inacessíveis para o Gil.

o	o	o	o	#	#	#	#
o	#	#	#	o	#	#	#
o	o	o	#	o	o	o	#
o	o	#	#	o	o	#	#
0	0	0	2	1	2	2	4

Por exemplo, na penúltima configuração, embora haja 3 gavetas fechadas à chave, só 2 estão inacessíveis, porque a gaveta de baixo pode ser acedida retirando a gaveta imediatamente acima dela. O Exemplo 1 abaixo corresponde a esta configuração.

Tarefa

Escreva um programa que, dada uma configuração da cómoda, calcula o número de gavetas inacessíveis para o Gil.

Input

A primeira linha tem um número inteiro, G, que é o número de gavetas da cómoda. Seguem-se G linhas com a configuração da cómoda, cada uma com um carácter que indica o estado da fechadura de uma gaveta. Esse carácter é "`#`", quando a gaveta está fechada à chave, ou a letra minúscula "`o`", quando a gaveta não está fechada à chave. A configuração da cómoda é

descrita do topo para a base: a primeira linha da configuração refere-se à gaveta de cima e a última linha refere-se à gaveta de baixo.

Restrições

$1 \leq G \leq 1000$ Número de gavetas da cómoda

Output

Uma única linha, com um inteiro que representa o número de gavetas inacessíveis para o Gil.

Exemplo 1

Input

```
4
#
#
o
#
```

Output

```
2
```

Exemplo 2

Input

```
9
#
o
#
#
#
o
#
#
o
```

Output

```
4
```

CSI Portugal

Aquela colega simpática que adora os livros e as séries policiais pediu-te ajuda, porque queria ter um programa para descobrir se um suspeito tem um álibi. A ideia dela até faz sentido: se um crime ocorreu no centro comercial entre as 14h e as 15h, alguém que esteve na escola das 9h às 16h não pode tê-lo cometido. Mas, se as provas apenas pudessem garantir que o crime tinha ocorrido entre as 14h e as 17h, a estada na escola das 9h às 16h já não seria um álibi, porque o suspeito poderia ter estado no centro comercial entre as 16h e as 17h.

Para ter um *álibi*, o suspeito tem de ter estado num local diferente do do crime durante todo o intervalo em que se estima que o crime ocorreu.

Tarefa

Escreva um programa que analise vários cenários independentes. Em cada cenário, são dados dois intervalos de tempo: um para o crime e o outro para uma atividade de um suspeito, realizada no dia do crime mas num local diferente. Pretende-se saber se, nesse cenário, o suspeito tem, ou não tem, um álibi.

Input

A primeira linha tem um inteiro, C, que representa o número de cenários. Por cada cenário, há duas linhas consecutivas. A primeira linha do cenário tem dois inteiros, I e J, que indicam que o crime ocorreu entre as I e as J horas de um dado dia num dado local. A segunda linha do cenário tem dois inteiros, S e T, que indicam que o suspeito esteve das S às T horas desse mesmo dia num local diferente.

Restrições

$1 \leq C \leq 100$	Número de cenários
$0 \leq I \leq J \leq 24$	Extremos do intervalo para um crime
$0 \leq S \leq T \leq 24$	Extremos do intervalo para a atividade de um suspeito

Output

Por cada cenário, há uma linha, com: "`Com alibi`", se a atividade do suspeito decorreu durante todo o intervalo estimado para o crime; "`Sem alibi`", nos restantes casos.

Exemplo

Input

```
4
14 15
9 16
14 17
9 16
0 5
0 8
19 24
19 19
```

Output

```
Com alibi
Sem alibi
Com alibi
Sem alibi
```

Tio Patinhas, o avarento?

O Tio Patinhas é considerado o pato mais rico do mundo. Para além de elevados montantes depositados em bancos e de muitas propriedades espalhadas por todo o mundo, ele mantém grande parte do seu dinheiro numa enorme caixa-forte na cidade de Patópolis.

Atualmente, o Tio Patinhas está sob um feitiço de amor e quer abdicar de toda a sua riqueza. Os seus sobrinhos-netos Huguinho, Zezinho e Luisinho consultaram uma feiticeira que lhes disse que a única maneira de quebrar o feitiço era convencer o tio-avô a colocar notas e moedas da sua caixa-forte num cofre especial. Mas, para a ação ter o efeito desejado, o valor total do dinheiro colocado no cofre teria de ser o maior possível, sem que o peso total desse dinheiro excedesse a capacidade do cofre (em gramas).

Salve o Tio Patinhas, indicando-lhe que notas e moedas da caixa-forte devem ser colocadas no cofre. Na caixa-forte só existem euros, podendo haver: notas de 500, 200, 100, 50, 20, 10 e 5 euros; moedas de 2 e 1 euros. Assuma que qualquer nota pesa 1 grama e qualquer moeda pesa 5 gramas.

Tarefa

Escreva um programa que, dados o peso máximo do dinheiro que pode ser colocado no cofre e as notas e as moedas que existem na caixa-forte, calcula o valor total, as notas e as moedas a colocar no cofre (maximizando o valor total sem exceder o peso máximo).

Input

A primeira linha contém dois inteiros: o peso P máximo do dinheiro que pode ser colocado no cofre (em gramas) e o número D de notas e moedas que existem na caixa-forte do Tio Patinhas. Seguem-se D linhas, cada uma com o valor V de uma nota ou moeda na caixa-forte. Se o valor V é 500, 200, 100, 50, 20, 10 ou 5, então é uma nota. Se o valor V é 2 ou 1, então é uma moeda.

Restrições

$1 \leq P \leq 2500$ Peso máximo do dinheiro no cofre

$1 \leq D \leq 500$ Número de notas e moedas na caixa-forte

Output

A primeira linha tem o valor total, em euros, do dinheiro a colocar no cofre.

As restantes linhas têm uma sequência com as notas e moedas a colocar no cofre. Cada uma destas linhas tem:

- o valor da nota (500, 200, 100, 50, 20, 10 ou 5) ou da moeda (2 ou 1); e
- a quantidade dessa nota ou moeda (um número inteiro positivo).

Estas linhas devem aparecer por ordem estritamente decrescente de valor da nota ou moeda, ou seja, primeiro aparece a nota de 500 euros (se for colocada alguma no cofre), depois a nota de 200 euros e assim sucessivamente.

Exemplo 1

Input

```
12 7
500
1
10
200
2
2
500
```

Output

```
1212
500 2
200 1
10 1
2 1
```

Exemplo 2

Input

```
12 3
2
1
2
```

Output

```
4
2 2
```

Exemplo 3

Input

```
4 2
1
2
```

Output

```
0
```

Exemplo 4

Input

```
8 4
200
1
2
10
```

Output

```
212
200 1
10 1
2 1
```

Magia com cartas

Queres aprender um truque de magia com cartas? Um dos mais habituais é pedir a uma pessoa para escolher uma carta de um baralho e depois adivinhar qual foi a carta escolhida. Isto não é difícil se o mágico souber a ordem das cartas e discretamente olhar para a carta seguinte à escolhida. Por exemplo, se o baralho estiver na ordem convencional e a carta seguinte for o 5 de ouros, a carta escolhida foi o 4 de ouros. Mas a ilusão só funciona se o mágico mostrar que as cartas estão desordenadas. Como saber a ordem das cartas e estas parecerem baralhadas?

Para começar, não há duas cartas do mesmo naipe juntas. Fixamos uma ordem para os naipes baseada numa mnemónica, como PECO, para Paus, Espadas, Copas e Ouros. O baralho poderia começar com uma carta de paus (P), seguida de uma carta de espadas (E), depois copas (C), ouros (O), paus (P), espadas (E) e assim sucessivamente. Quando se acabam os naipes na sequência PECO, volta-se ao início.

Agora vamos supor que os valores das cartas também seguem um padrão cíclico. Considerando que o ás é representado por 1, o valete por 11, a dama por 12, o rei por 13 e as restantes cartas são representadas pelo respetivo número, a sequência de valores das cartas poderia ser: $1, 2, \ldots, 13, 1, 2$, etc. Conciliando esta regra com a ordem PECO para os naipes, a ordenação do baralho seria:

1 P, 2 E, 3 C, 4 O, 5 P, 6 E, 7 C, 8 O, 9 P, 10 E, 11 C, 12 O, 13 P, 1 E, 2 C, ...

Mas como os valores seguidos dão muito nas vistas, podemos saltar de 3 em 3. Com *salto* 3, a ordem do baralho seria 1 P, 4 E, 7 C, 10 O, 13 P, 3 E e assim sucessivamente.

A ordem PECO é perfeitamente arbitrária. Poderia ser CEPO, ou qualquer outra permutação das letras P, E, C e O. O salto também pode ser qualquer número entre 2 e 12 (porque o número de cartas de cada naipe, 13, é primo). Ou seja, fixando uma sequência de naipes e um salto, consegue-se definir uma ordem para as cartas do baralho que cria a ilusão de desordem aleatória. Nota que a ordem é preservada se se partir o baralho em dois e a metade que estava por cima for colocada no fundo. Claro que não se pode baralhar realmente as cartas. No entanto, se se mostrar o baralho aberto em leque, parece que as cartas foram baralhadas. Deixar um espectador partir o baralho aumenta a ilusão porque parece que este influiu na ordem das cartas.

Tarefa

Escreva um programa que, dada uma ordem para os naipes e um salto, determine, para uma carta, qual a anterior na ordem definida.

Input

A primeira linha tem uma sequência com 4 caracteres, com as letras `P`, `E`, `C` e `O`, por qualquer ordem. A segunda linha tem um inteiro S, que é o valor do salto. Seguem-se k linhas, constituídas por um inteiro V seguido de um carácter N, representando uma carta do baralho. O inteiro V é o valor da carta. O carácter N é uma letra maiúscula (`P`, `E`, `C` ou `O`) que representa o naipe. A última linha tem "`0`" (zero).

Restrições

$2 \leq S \leq 12$ Salto

$1 \leq k \leq 20$ Número de cartas

$1 \leq V \leq 13$ Valor de uma carta

Output

Uma linha por cada carta introduzida, com a carta anterior na ordem definida. As cartas são representadas no mesmo formato do input.

Exemplo 1

Input

```
PECO
3
5 O
13 E
1 C
3 P
0
```

Output

```
2 C
10 P
11 E
13 O
```

Exemplo 2

Input

```
CEPO
5
5 O
13 E
1 C
3 P
4 C
7 E
0
```

Output

```
13 P
8 C
9 O
11 E
12 O
2 C
```

Joaquim Moedas

Joaquim Moedas: *O seu troco, Dona Graça: 13 moedas de um cêntimo, 13 moedas de dois cêntimos e 13 moedas de cinco cêntimos, ou seja, um euro e quatro cêntimos.*

Graça Desgraçada: *Ó Santa Engrácia! Cinco minutos à espera para isto? Não me desgrace com tantas moedas, Sr. Joaquim!!*

Joaquim Moedas: *Calma, Dona Graça. Um dia ainda vai engraçar com estas coisas... Prefere 8 moedas de um cêntimo, 8 moedas de dois cêntimos e 8 moedas de dez cêntimos? Ou 4 moedas de um cêntimo, 4 moedas de cinco cêntimos e 4 moedas de vinte cêntimos? É que não lhe posso dar 1,04€ com mais de* ***três tipos*** *diferentes de moedas, dando-lhe o* ***mesmo número*** *de moedas* ***de cada tipo****.*

Joaquim Moedas, dono da única mercearia de Capareiros, tem uma *trocopatia* que não passa despercebida a ninguém: só dá trocos com o maior número de **tipos de moedas** (i.e., moedas de valores distintos) que permite totalizar o valor a devolver usando um **número igual** de moedas **de cada tipo**.

É possível perfazer 1,04€ com moedas de seis tipos: 1 moeda de cinquenta cêntimos, 1 de vinte cêntimos, 2 de dez cêntimos, 2 de cinco cêntimos, 1 de dois cêntimos e 2 de um cêntimo. Mas o Sr. Joaquim nunca daria um troco destes porque o número de moedas de cada tipo não é igual. Para dar 1,04€, o Sr. Joaquim usa três tipos de moedas (três é o maior número possível de tipos usando o mesmo número de moedas de cada tipo) e tem três maneiras de fazer o troco:

- usando moedas de 1, 2 e 5 cêntimos (13 de cada);
- usando moedas de 1, 2 e 10 cêntimos (8 de cada);
- usando moedas de 1, 5 e 20 cêntimos (4 de cada).

Uns zombam, outros estranham (e até receiam o contágio); mas todos os Capareirenses partilham uma preocupação: O Sr. Joaquim demora uma eternidade a determinar com que moedas vai dar o troco. Podem-no ajudar a ser mais rápido?

Tarefa

Escreva um programa que, dado o valor do troco, determine com quantos tipos de moedas o Sr. Joaquim daria esse troco e quantas maneiras tem de o fazer. Os tipos de moedas a considerar no troco são os do sistema Euro: 1c, 2c, 5c, 10c, 20c 50c, 1EUR e 2EUR.

Input

A primeira linha tem um inteiro, N, que representa o número de trocos a analisar. Seguem-se N linhas, cada uma com um número inteiro positivo, V, que representa o valor do troco em cêntimos de euro.

Restrições

$1 \leq N \leq 1000$ Número de trocos
$1 \leq V < 2^{31}$ Valor de um troco

Output

N linhas, uma por cada troco, com dois valores, T e M, separados por um espaço. T é o número de tipos de moedas que o Sr. Joaquim daria no troco (é o número máximo de tipos de moedas com o qual se pode dar o troco usando o mesmo número de moedas de cada tipo). M é o número de maneiras que o Sr. Joaquim tem para fazer o troco.

Exemplo 1

Input

```
9
104
20
200
10
1
100
5
2
50
```

Output

```
3 3
1 5
3 1
1 4
1 1
2 1
1 2
1 2
2 1
```

Exemplo 2

Input

```
5
4
16
18
32
2147483646
```

Output

```
1 2
3 2
4 1
3 3
6 1
```

Ainda os apanhamos

No ToPAS, à semelhança do *ACM International Collegiate Programming Contest*, as equipas são classificadas pelo número de problemas resolvidos. Para desempatar, usa-se a soma dos tempos dos problemas resolvidos (menor soma, melhor classificação). O tempo de um problema é igual ao tempo que decorreu desde o início do concurso até à sua submissão correta, acrescido de uma penalização de 20 minutos por cada submissão errada para esse problema. Não havendo mais critérios de desempate, todas as equipas que ficarem empatadas no primeiro lugar são *vencedoras*.

As equipas têm acesso ao *ranking* global durante a prova mas, para criar algum suspense, pode haver *blackout* na parte final do concurso. No período de *blackout*, cada equipa conhece o resultado de todas as suas submissões, mas não tem informação sobre as submissões efetuadas pelas outras equipas. Por isso, mesmo que uma equipa tenha resolvido todos os problemas, pode não saber se é vencedora.

O concurso tem **sete** problemas, dura **quatro** horas e há *blackout*, que começa na última hora da prova, pelo menos a 10 minutos do fim. As submissões (e o início do *blackout*) ocorrem num *instante*, que corresponde ao número de segundos decorridos desde o início da prova. Uma submissão que ocorra no instante inicial do *blackout* já não é visível pelas outras equipas. Impomos um intervalo com duração mínima de D segundos entre submissões consecutivas de uma mesma equipa. Assim, se uma equipa efetuar uma submissão no instante 100 e D for 15, essa equipa só pode voltar a submeter no instante 115, independentemente de ser uma submissão para o mesmo problema ou para outro. Nenhuma equipa pode efetuar submissões para problemas que já tenha resolvido.

Tarefa

Escreva um programa para descobrir se uma equipa V é vencedora, sabendo que essa equipa resolveu os sete problemas da prova, seis dos quais antes do *blackout*, e que nenhuma equipa completou os sete problemas antes do *blackout*. O programa analisará todas as submissões feitas antes do *blackout* e as submissões da equipa V durante o *blackout*.

Input

A primeira linha tem cinco inteiros, E, S, D, B e V, que definem o número de equipas, o número de submissões antes do *blackout*, a duração mínima (em segundos) do intervalo entre submissões de uma mesma equipa, o instante inicial do *blackout* e o identificador da equipa que pretende saber se pode ser vencedora. Seguem-se S linhas, cada uma com quatro inteiros que descrevem uma submissão: o identificador da equipa que a fez, o instante t em que ocorreu, o identificador do problema e o resultado (0 se não foi aceite e 1 se foi aceite). Os problemas são identificados por inteiros de 1 a 7 e as equipas por inteiros de 1 a E.

Depois, há uma linha com um inteiro K que indica o número de submissões da equipa V

durante o *blackout*, para o problema em falta. Seguem-se K linhas, cada uma com um inteiro t que define o instante da submissão correspondente (a última foi aceite).

Restrições

$5 \leq E \leq 100$	Número de equipas
$6 \leq S \leq 10\,000$	Número de submissões antes do *blackout*
$10 \leq D \leq 30$	Duração mínima do intervalo entre submissões de uma equipa
$10\,800 \leq B \leq 13\,800$	Instante inicial do *blackout*
$1 \leq K \leq 60$	Número de submissões da equipa V durante o *blackout*
$0 < t \leq 14\,400$	Instante de uma submissão

Output

Uma linha com uma das frases "`Vencemos`" ou "`Nao sabemos`" (sem acentos), que representa a conclusão. Se puder haver empate no primeiro lugar, também escreverá "`Vencemos`".

Exemplo 1

Input

```
7 24 30 12600 1
1 300 1 1
2 420 1 0
2 600 3 0
1 1800 2 1
2 1830 1 1
2 1950 3 1
4 2400 1 1
3 2520 2 1
2 3000 4 0
4 4500 3 1
1 7200 3 0
2 7215 4 1
2 8100 5 1
1 8400 6 1
5 8500 1 1
2 10800 6 1
1 10810 7 1
2 11100 7 0
5 11100 2 0
1 11400 5 1
1 11460 3 1
2 11700 2 1
1 12300 4 0
2 12599 7 0
1
13655
```

Output

```
Nao sabemos
```

Exemplo 2

Input

```
7 23 30 12600 1
1 300 1 1
2 420 1 0
2 600 3 0
1 1800 2 1
2 1830 1 1
2 1950 3 1
4 2400 1 1
3 2520 2 1
2 3000 4 0
4 4500 3 1
1 7200 3 0
2 7215 4 1
2 8100 5 1
1 8400 6 1
5 8500 1 1
2 10800 6 1
1 10810 7 1
2 11100 7 0
5 11100 2 0
1 11400 5 1
1 11460 3 1
2 11700 2 1
2 12599 7 0
2
12803
13654
```

Output

```
Vencemos
```

Exemplo 3

Input

```
7 23 30 12600 1
1 300 1 1
2 420 1 0
2 600 3 0
1 1800 2 1
2 1830 1 1
2 1950 3 1
4 2400 1 1
3 2520 2 1
2 3000 4 0
4 4500 3 1
1 7200 3 0
2 7215 4 1
2 8100 5 1
1 8400 6 1
5 8500 1 1
2 10800 6 1
1 10810 7 1
2 11100 7 0
5 11100 2 0
1 11400 5 1
1 11460 3 1
2 11700 2 1
2 12599 7 0
3
12803
13201
13654
```

Output

```
Nao sabemos
```