Problema A

Título de eleitor

Nome do arquivo fonte: `eleitor.c`, `eleitor.java`, `eleitor.cpp` *ou* `eleitor.pas`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Neste problema, necessita-se validar números de título de eleitor, que são dados na forma 9999999999-99. Um número é considerado válido se os seus dois últimos dígitos, chamados dígitos verificadores (DV) forem iguais aos dígitos calculados segundo as regras apresentadas a seguir.

a) O número do título de eleitor é composto por 12 dígitos numéricos (descontando-se então o hífen que separa os DVs do restante do número), devendo ser preenchido com zeros à esquerda para completar essa quantidade caso seja necessário:

```
0044562101-23   2345664201-22   0000012202-19
```

b) A verificação do título de eleitor é feita em duas etapas, uma para determinar o primeiro dígito verificador (DV1) e outra para calcular o segundo (DV2).

c) Cálculo do primeiro Dígito Verificador (DV1):

- Multiplica-se o primeiro dígito (o mais à esquerda) por 9, o segundo por 8, o terceiro por 7 e assim sucessivamente até o oitavo dígito, que deve ser multiplicado por 2, e guarda-se a soma dos resultados dessas multiplicações.

  σ = (1º Dígito * 9) + (2º Dígito * 8) + ... + (8º Dígito * 2)

- Em seguida, calcula-se o resto da divisão dessa soma por 11 e, se ele for maior que 1, o primeiro dígito verificador é dado pela subtração: 11 – resto. Caso o resto calculado anteriormente seja igual a 0 ou a 1, deve-se verificar se o nono e décimo dígitos do título são iguais a ‘01’ ou se, em vez disso, o décimo dígito é igual a ‘2’. Caso seja, DV1 é o inverso do resto anteriormente calculado: se resto for zero, DV1 = 1; se resto for um, DV1 = 0. Nos casos restantes o primeiro dígito verificador é igual a zero.

d) Cálculo do segundo Dígito Verificador (DV2):

- Multiplica-se o nono dígito por 4, o décimo dígito por 3 e o primeiro dígito verificador recém calculado por 2, soma-se os resultados dessas multiplicações e calcula-se o resto da divisão dessa soma por 11.

- Se esse resto for maior que 1, o segundo dígito verificador é dado pela subtração: 11 – resto. Caso, entretanto, o resto seja igual a 0 ou a 1, deve-se verificar se o nono e décimo dígitos do título são iguais a ‘01’ ou se, em vez disso, o décimo dígito é igual a ‘2’. Caso seja, DV2 é o inverso do resto anteriormente calculado: se resto for zero, DV2 = 1; se resto for um, DV2 = 0. Nos casos restantes, o segundo dígito verificador é igual a zero.

Se a concatenação de DV1 com DV2 for igual à concatenação do décimo primeiro dígito com o décimo segundo, o número do título de eleitor é válido, senão é inválido.

Entrada

O conjunto de casos de teste é composto por diversos números de título de eleitor, cada um contendo até 13 caracteres, sendo o antepenúltimo caractere sempre um hífen e os restantes,

dígitos de 0 a 9. O programa deverá encerrar seu processamento quando for lida a entrada 0000000000-00.

Saída

Para cada caso de teste, você deve imprimir uma linha com o texto ‘correto’, em minúsculas, se o número de título de eleitor for válido segundo as regras descritas anteriormente ou, caso seja inválido, imprimir o par de dígitos verificadores corretamente calculado pelo programa, sem espaços ou qualquer outro tipo de separador entre eles.

Exemplo de entrada

```
44562101-16
2345664201-67
0044562101-23
71805401-16
12202-81
2316905101-16
2345664201-22
12202-19
0000000000-00
```

Exemplo de Saída

```
correto
correto
16
32
correto
75
67
81
```

Problema B

Revisão de Código de Barras de Boleto

Nome do arquivo fonte: `boleto.c`, `boleto.java`, `boleto.cpp` *ou* `boleto.pas`

Autor: Daniel Facciolo Pires (Fatec Franca)

Durante anos, todos os boletos do Banco dos Munhequeiros MB (BM) foram digitados em um velho teclado de computador. Recentemente, o Sr. Jeferson, um dos bancários do BM, percebeu que o teclado apresentava falha em uma, e apenas uma, das teclas numéricas. Mais especificamente, a tecla com defeito, quando pressionada, não insere o respectivo dígito numérico na folha, como se não tivesse sido pressionada. O Sr. Jeferson percebeu que isso poderia ter alterado os valores numéricos representados nos códigos de barra dos boletos e, preocupado com o fechamento do caixa, quer saber, a partir dos valores originais negociados nos boletos que ele mantinha em anotações manuscritas, quais os valores de fato representados nos códigos de barra dos boletos. Por exemplo, se o teclado apresenta falha no dígito 5, o valor 1500 seria impresso no boleto como 100, pois o 5 não seria impresso. Note que o Sr. Jeferson quer saber o valor ***numérico*** representado no boleto, ou seja, nesse mesmo computador, o número 5000 corresponde ao valor numérico 0, e não 000 (como ele de fato aparece impresso).

Entrada

A entrada consiste de diversos casos de teste, cada um em uma linha. Cada linha contém dois inteiros D e N ($1 \leq D \leq 9$, $1 \leq N \leq 10^{100}$) separados por um espaço em branco, representando, respectivamente, o dígito que está apresentando problema no teclado e o número que foi negociado originalmente no boleto. O último caso de teste é seguido por uma linha que contém apenas dois zeros separados por um espaço em branco.

Saída

Para cada caso de teste da entrada, o seu programa deve imprimir uma linha contendo um único inteiro V, o valor numérico representado de fato no código de barras do boleto.

Exemplo de entrada

```
5 5000000
3 123456
9 23456867153
9 99999999911199
7 777
0 0
```

Exemplo de Saída

```
0
12456
23456867153
111
0
```

Problema C

Sonâmbulos

Nome do arquivo fonte: `sonambulos.c`, `sonambulos.java`, `sonambulos.cpp` *ou* `sonambulos.pas`

Autor: Daniel Facciolo Pires (Fatec Franca)

Antigamente tudo funcionava bem na questão de como os sonâmbulos saem dos corpos enquanto dormem. Para cada pessoa que dorme em sua localidade, o respectivo sonâmbulo fica sob a responsabilidade de uma divindade local que possui o seu próprio reino dos sonâmbulos. Com o advento da espiritualidade, as diversas divindades começaram a entrar em conflito sobre quem tinha responsabilidade por cada sonâmbulo. A situação piorou quando algumas divindades começaram a reclamar que estavam sobrecarregadas, principalmente os da região de cidades mais populosas no planeta Terra. Para resolver a situação de uma vez por todas, realizou-se uma reunião em uma pizzaria na Itália com os representantes de assuntos de sonambulismo das principais divindades existentes:

- Pedro, líder dos sonâmbulos na Ásia e Oceania;
- Mara, matriarca dos sonâmbulos na Europa;
- Carlos, guia dos sonâmbulos na América;
- Zeca, juiz dos sonâmbulos na África.

Depois de muita discussão, grande confusão a respeito de termos dos sonos e dificuldade para entender as várias línguas e dialetos, chegou-se ao seguinte acordo: a partir de uma certa data, os sonâmbulos de todos os seres viventes seriam divididos igualmente entre os reinos dos sonâmbulos, pela ordem em que dormiam. O primeiro a dormir iria para o Carlos, o segundo para Zeca, o terceiro para o Pedro, o quarto para a Mara, o quinto para o Carlos, e assim sucessivamente, nesta ordem.

Dada a ordem que uma criatura dormisse desde a data de validade do acordo, escreva um programa que decida qual reino do além será seu destino.

Entrada

Cada caso de teste terá uma linha com um inteiro *N*, variando de 1 a 2000000000, representando a ordem que uma pessoa entrou em estado de sonambulismo desde a data do firmamento do acordo. Encerrar o processamento quando for lida uma entrada contendo o número 0.

Saída

Para cada caso de teste, imprimir uma linha contendo o destino da alma: Pedro, Mara, Carlos ou Zeca, com a inicial em maiúsculas e o restante em minúsculas.

Exemplo de entrada

```
3
24846
1000000
0
```

Exemplo de Saída

```
Pedro
Zeca
Mara
```

Problema D

Bolsa de Valores

Nome do arquivo fonte: `bolsa.c`, `bolsa.java`, `bolsa.cpp` *ou* `bolsa.pas`

Autor: Leandro Luque (Fatec Mogi das Cruzes)

Darcy está procurando uma fonte alternativa de renda como complemento para o seu salário. Conversando com amigos, ele se convenceu de que investir na Bolsa de Valores seria uma opção interessante. Por ser muito cauteloso, Darcy decidiu fazer um estudo da variação do valor das ações de diversas empresas no decorrer de alguns meses. Entre os dados que tem registrado, está a identificação do período no qual a variação acumulada do valor das ações de uma empresa tenha sido máxima.

Para agilizar o estudo, Darcy o convidou para desenvolver um programa de computador que, dadas as variações mensais no valor das ações de uma empresa, retorna o período (mês inicial e final) no qual a variação acumulada seja máxima. Como pagamento pelos seus serviços, ele garantiu uma participação nos lucros, caso seu plano de investimento dê certo.

Entrada

A entrada do programa é composta por diversos casos de teste. A primeira linha de cada caso de teste contém um número inteiro *N* representando a quantidade de meses (variando de 1 a 200) para os quais Darcy registrou a variação no valor das ações de uma empresa. Cada uma das *N* linhas seguintes contém 1 número inteiro (variando de -100 a 100) indicando uma variação no valor das ações para um mês (o 1º número corresponde à variação no 1º mês; o 2º número corresponde à variação no 2º mês; e assim sucessivamente). O último caso de teste é seguido por uma linha que contém um zero.

Saída

Para cada caso de teste, imprima dois números inteiros, separados por espaço, representando o mês inicial e final no qual a variação das ações é máxima (o 1º mês é numerado como 1). Se mais de uma combinação de meses resultar na mesma variação máxima para um caso de teste, retorne aquela que ocorreu por último, considerando primeiro o mês inicial e, em seguida, o mês final.

Exemplo de entrada

```
8
20
-30
15
-10
30
-20
-30
30
0
```

Exemplo de Saída

```
3 5
```

Problema E

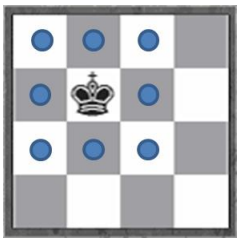
Minixadrez

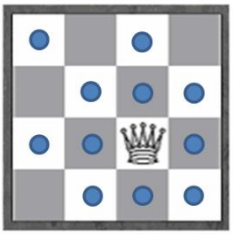
Nome do arquivo fonte: `minixadrez.c`, `minixadrez.java`, `minixadrez.cpp` *ou* `minixadrez.pas`

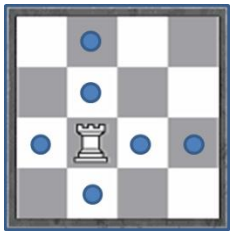
Autor: Leandro Luque (Fatec Mogi das Cruzes)

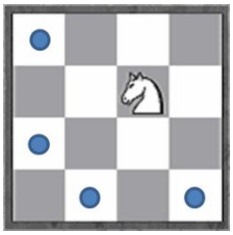
Xadrez é um jogo de tabuleiro muito antigo no qual dois jogadores procuram colocar o rei adversário em um estado conhecido como "xeque-mate". Neste estado, o rei adversário está sob ataque e, para qualquer movimento que faça em uma jogada, continuará sob ataque. Quando isso ocorrer, o adversário perderá a partida. Manuela e João gostam de jogar uma versão simplificada do xadrez, conhecida como minixadrez. Nesta versão, o tabuleiro possui apenas 4 linhas e 4 colunas, além de existirem apenas 4 tipos de peças: rei, rainha, torre e cavalo.

O movimento das peças segue as mesmas regras da versão tradicional: o rei pode mover-se para qualquer casa adjacente; a rainha pode mover-se qualquer quantidade de casas na vertical, horizontal e nas diagonais; a torre pode mover-se qualquer quantidade de casas na vertical ou horizontal; e o cavalo move-se em *L*, conforme a figura seguinte - os pontos na figura representam as possíveis casas para onde a peça pode mover-se em uma jogada.


Rei


Rainha


Torre


Cavalo

Como exemplo de posição de xeque-mate para o minixadrez, considere a figura a seguir. Nela, o rei preto está sendo atacado pela rainha branca e, para qualquer casa que se mova, continuará sob ataque das brancas - o rei não pode capturar a rainha, pois o cavalo a protege.

Manuela tem estudado bastante para ganhar de João e recentemente está se concentrando nas jogadas que podem colocar o rei do João em xeque-mate. Sua estratégia de estudo consiste inicialmente em posicionar o rei preto do João em uma das casas do tabuleiro. Em seguida, ela posiciona uma rainha, uma torre e um cavalo brancos em outras casas do tabuleiro, de tal forma que o rei preto ainda não esteja sendo atacado. Por fim, ela tenta encontrar um único movimento de suas peças que coloque o rei do João em xeque-mate.

Como ainda está insegura quanto a sua análise de jogo, Manuela pediu para que você desenvolvesse um programa de computador que, dadas as posições das peças, determina se é possível fazer um movimento que coloque o rei preto em xeque-mate, bem como qual é esse movimento.

Entrada

A entrada do programa é composta por diversos casos de teste. Cada caso de teste é especificado por 4 linhas, cada uma delas contendo 2 números inteiros (variando de 1 a 4 cada) separados por espaço. Os números representam a linha e coluna, respectivamente, onde se encontra uma peça – a casa do canto superior esquerdo é 1 1. A primeira linha especifica a posição do rei preto. A segunda, a posição da rainha branca. A terceira, a posição da torre branca; e a quarta, a posição do cavalo branco. Para um mesmo caso de teste, nunca existirão duas peças na mesma casa. Nenhum caso de teste contém um caso no qual o movimento de mais de uma peça branca possa resultar em xeque-mate, ou seja, se o xeque-mate for possível, apenas uma delas, se movida, poderá colocar o rei preto em xeque-mate. O último caso de teste é seguido por uma linha que contém dois zeros separados por espaço.

Saída

Para cada caso de teste, imprima o caractere ‘N’, caso não seja possível colocar o rei preto em xeque-mate em apenas uma jogada. Caso contrário, imprima o caractere representativo da peça branca que deverá ser movida - ‘R’, para rainha; ‘T’, para torre; e ‘C’, para cavalo – separado por um espaço do número da linha (variando de 1 a 4), separada por um espaço do número da coluna (variando de 1 a 4), para a qual a peça especificada deverá ser movida. Todos os caracteres alfabéticos deverão ser impressos em maiúsculas e sem aspas.

Exemplo de entrada

```
2 2
4 3
4 1
4 4
1 4
3 1
4 1
4 2
0 0
```

Exemplo de Saída

```
R 2 3
N
```

Problema F

Saldo médio da conta

Nome do arquivo fonte: `saldo.c`, `saldo.java`, `saldo.cpp` *ou* `saldo.pas`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Tibério está pretendendo casar. E quem casa, quer casa, como diz o velho ditado. Mas Tibério ainda não tem casa e, por isso, deseja comprar uma. Financiada, obviamente. Consultando o seu gerente no banco, ele soube que este utiliza o saldo médio da conta-corrente da pessoa para determinar o valor a ser cobrado pelo financiamento. Assim, Tibério está interessado em calcular o seu e você, o amigo programador do nosso herói, recebeu a incumbência de criar um programa que faça isso pra ele. A partir de um conjunto de lançamentos ocorridos na conta corrente, e considerando o saldo inicial da conta no período, será possível determinar o saldo ao final de cada dia. Depois basta determinar a média do período, que é a média aritmética simples dos saldos diários.

Um lançamento pode ser um crédito, ou seja, dinheiro entrando na conta através de algum tipo de depósito, ou então poderá ser um débito, significando que uma quantia foi retirada da conta. Nos créditos, o saldo da conta aumenta, nos débitos ele diminui. Como uma conta pode ter vários lançamentos de cada tipo em um dia, o saldo final do dia D é dado pelo saldo da conta no final do dia D-1 (ou seja, o dia anterior) mais a soma dos créditos do dia D menos a soma dos débitos desse dia D.

Entrada

Existem vários casos de teste, cada um correspondendo a um período para o qual o saldo médio deverá ser calculado. Cada caso de teste é iniciado por um inteiro positivo $Q < 500$ que indica a quantidade de dias do período, um inteiro $L > 0$ e ≤ 1000, que corresponde à quantidade de lançamentos ocorridos no período e um real S que informa o saldo da conta no início do período a ser considerado. Seguem-se então L lançamentos, cada um composto por um inteiro positivo $D \leq Q$, que indica o dia do período (1 é o primeiro dia do período, 2 o segundo, etc) em que houve o lançamento, um caracter T, que pode ser 'C' (em maiúsculas) para 'Crédito' ou 'D' (em maiúsculas) para 'Débito', informando o tipo do movimento e um real V, indicando o valor daquela movimentação. Você pode assumir que os lançamentos são informados sempre em ordem crescente do dia: os lançamentos mais antigos vêm antes dos lançamentos mais recentes. As entradas se encerram quando for lido um Q igual a zero.

Saída

Para cada caso de teste você deve imprimir uma linha indicando o número do caso de teste, conforme o exemplo a seguir, seguido por dois pontos e o saldo médio computado para o período, com duas casas depois da vírgula. OBS: utilize variáveis de ponto flutuante de precisão dupla nos seus cálculos.

Exemplo de entrada

```
45 10 102.30
1 C 500.00
7 D 82.90
7 D 122.47
9 C 103.25
19 D 52.32
22 D 129.44
29 C 320.00
35 D 97.23
```

```
44 D 129.39
45 D 30.45
10 3 100.00
2 D 10.00
4 D 10.00
8 D 10.00
5 2 100.00
2 D 120.00
3 D 40.00
0 0 0.00
```

Exemplo de Saída

```
Caso 1: 499.47
Caso 2: 81.00
Caso 3: -20.00
```

Problem G

Seasons

Source file name: `seasons.c`, `seasons.java`, `seasons.cpp` *or* `seasons.pas`

Authors: Antonio Cesar de Barros Munari / Paulo Edson (Fatec Sorocaba)

We are all familiar with the ancient practice of dividing the year in periods commonly named seasons. Their quantity, name, period of the year and properties are dependent of cultural, historical and geographic issues, but their roles are the same for all the people in the world. In Western Europe, for example, four seasons are recognized:

- Spring, the flowers season, starts March 21st and ends June 20th;
- Summer, the hot season, starts June 21st and ends September 22nd;
- Autumn, the falling season, starts September 23rd and ends December 20th; and
- Winter, the cold season, starts December 21st and ends March 20th.

Yes, we know for sure that the dates on which the seasons really start and finish could vary a little from year to year, and also depend on the exact hour of the day, but we could assume the approximation presented above. In this problem, maybe the easiest task of this contest (or not …, you may disagree, of course), is to ask you to develop a very simple program to determine what season a day belongs to.

Input

The input consists of a various entries. The first row has a positive integer $N \leq 400$ indicating how many dates must be processed. There are N rows, each one containing 3 positive integers > 0: $D \leq 31$, $M \leq 12$ and $Y \leq 2020$, the day, month and year of that date, respectively. Assume that all dates in the input are valid ones.

Output

For each date in the input your program should output the correct name of the corresponding season for the western European people, according to the previous explanation in the description of this problem. The name of the season must be printed in lower case with a break line after it. No spaces, tabs or other special characters should be printed.

Sample input

```
7
19 5 2012
1 3 2010
22 3 2015
20 3 2015
21 3 2015
30 6 2010
9 10 2018
```

Sample output

```
spring
winter
spring
winter
spring
summer
autumn
```

Problema H

SPQR

Nome do arquivo fonte: `spqr.c`, `spqr.java`, `spqr.cpp` *ou* `spqr.pas`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Paulo está estudando um conjunto de documentos muito antigos que ele encontrou pela Web em uma biblioteca on-line e está meio atrapalhado com a numeração romana utilizada. Como você já deve estar imaginando, esses documentos são realmente muito velhos, da época do Império Romano, e parecem compor uma espécie de recenseamento populacional, indicando o tamanho de agrupamentos militares existentes na época. Paulo dispõe da identificação dos soldados e, portanto, consegue contar quantos existem em cada unidade, mas, para conferir o seu resultado com o dos documentos originais, ele precisa converter seu número em base decimal para a correspondente notação de numeração romana. Em geral ele faz somas de valores em notação decimal e precisa do resultado correto em numeração romana. Você deverá ajudá-lo, criando um programa que recebe pares de números em notação decimal e produz o resultado da soma desses números em romano. Considere que a numeração romana convencional abrange os valores de 1 até 3999 apenas. Lembre-se também que I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 e M = 1000 e que existem combinações aditivas, como, por exemplo, VI = 6, VII = 7, XXXVIII = 38, e também subtrativas, dadas pelos dígrafos IV = 4, IX = 9, XL = 40, XC = 90, CD = 400 e CM = 900. Isso ocorre porque não é possível utilizar o mesmo dígito mais de 3 vezes seguidas e, assim, 400 não pode ser escrito como CCCC e sim como CD, ou seja, 500 menos 100, por exemplo. Considere que todo número decimal tem apenas uma representação romana convencional válida.

Entrada

O conjunto de casos de teste é iniciado por um inteiro positivo $N \leq 10000$ que indica a quantidade de operações de soma a serem realizadas. Seguem-se N pares de inteiros X e Y, $0 < X, Y < 4000$.

Saída

Para cada caso de teste, você deve imprimir uma linha indicando apenas o resultado da soma, em numeração romana. Caso o resultado esteja fora do intervalo válido da numeração romana convencional, imprimir uma sequência contendo exatamente cinco asteriscos.

Exemplo de entrada

```
6
1 1
3999 3
112 187
1024 512
2020 2020
999 333
```

Exemplo de Saída

```
II
*****
CCXCIX
MDXXXVI
*****
MCCCXXXII
```

Problema I

Esteira ergométrica

Nome do arquivo fonte: `esteira.c`, `esteira.java`, `esteira.cpp` *ou* `esteira.pas`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Felipe começou a malhar, pois percebeu que estava sem energia para as diversas atividades do dia-a-dia. Como está ainda nas primeiras semanas de academia, tem passado muito tempo na esteira ergométrica, para melhorar sua capacidade cardiovascular e queimar seus quilos extras mais rapidamente. A esteira que ele usa é cheia de recursos, permitindo uma série de ajustes e controles de desempenho, e possui um painel parecido com o da figura 1. A programação mais comum que Felipe tem utilizado requer que ele informe um tempo de duração da atividade em minutos e qual a velocidade inicial da esteira, dada em km/h. O equipamento então começa a funcionar naquela velocidade informada e, depois de algum tempo, acelera para uma velocidade 3km/h mais rápida que a inicial, permanecendo uma certa quantidade de tempo nesse estado. Em seguida, volta a funcionar um novo período de tempo na velocidade inicial e depois alterna novamente para a velocidade mais alta, fazendo essas trocas periodicamente até que o tempo total da atividade seja atingido. Isso obriga Felipe a se exercitar em velocidades diferentes, melhorando seu gasto de energia e fortalecendo seu sistema cardiovascular. Como ficar 50 minutos sobre uma esteira é tedioso, nosso atleta fica tentando ocupar sua mente com questões lógicas, e esse painel permite uma série de divagações.

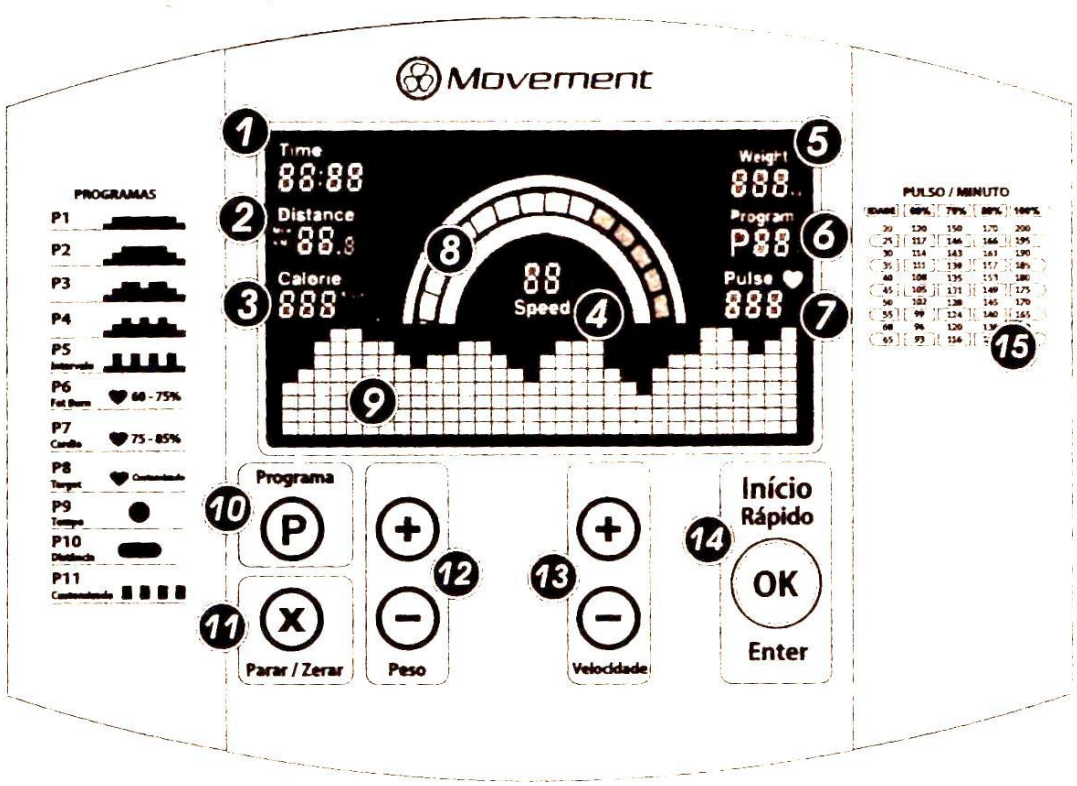


Figura 1: Painel digital da esteira ergométrica utilizada por Felipe.

O item 9 da figura é um gráfico de 32 colunas que representa a variação da velocidade da esteira, e uma coluna mais alta representa um período mais veloz do que uma coluna mais baixa. Se a atividade for configurada para ter, por exemplo, 32 minutos de duração, cada coluna corresponderá exatamente a 1 minuto, se a atividade tiver 16 minutos de duração, cada coluna corresponderá a 30 segundos, e assim por diante. A programação que Felipe usa

alterna sempre entre duas velocidades, a informada inicialmente e outra que é 3km/h mais rápida. Ele deseja saber, dada a configuração das colunas, a duração total da atividade e a velocidade inicial informada, qual a distância total percorrida e qual a velocidade média desenvolvida sobre a esteira.

Entrada

O conjunto de casos de teste é composto por diversas entradas. Inicialmente um inteiro N é informado, indicando a quantidade de casos de teste. Em seguida seguem-se N*2 linhas, descrevendo os casos de teste, cada um representando uma situação a ser avaliada. Cada caso de teste possui uma primeira linha composta por um inteiro T ($0 < T \leq 99$) indicando a duração total da atividade e um real V ($0 < V \leq 25$) indicando a velocidade inicial da esteira. A segunda linha do caso de teste possui 32 caracteres representando cada uma das colunas do mostrador do painel digital da esteira. Nessa sequência um caractere 'b' (em minúscula) representa a velocidade mais baixa, que é a velocidade inicialmente informada por Felipe ao configurar a esteira, e um caractere 'A' (em maiúscula) corresponde à velocidade mais alta, que é sempre 3km/h superior à velocidade mais baixa.

Saída

Para cada caso de teste você deve imprimir uma linha com dois números reais arredondados para uma casa decimal indicando respectivamente a distância total percorrida e a média de velocidade desenvolvida.

Exemplo de entrada

```
3
32 7.0
bbAAAAbbbbAAAAbbbbAAAAbbbbAAAAbb
16 5.0
bbAAAbbbAAAbbbAAAbbbAAAbbbAAAbbb
25 6.5
bbAAAAAbbbbbbAAAAAbbbbbbAAAAAbbb
```

Exemplo de Saída

```
4.5 8.5
1.7 6.4
3.3 7.9
```

Problema J

Festa

Nome do arquivo fonte: `festa.c`, `festa.java`, `festa.cpp` *ou* `festa.pas`

Autor: Leandro Luque (Fatec Mogi das Cruzes)

Joãozinho é muito conhecido pelas festas que organiza, quase sempre muito divertidas e fartas de aperitivos e bebidas saborosas. No entanto, sua fama ficou um pouco abalada após sua última festa. Como poucos dos convidados se conheciam, não houve muita interação e ela acabou terminando muito cedo. Procurando recuperar sua fama, Joãozinho está decidido a não repetir o mesmo erro e, para a próxima festa, deseja convidar pessoas que conheçam no mínimo *N* outros convidados.

Ajude Joãozinho a escrever um programa de computador que, dada uma lista de pessoas e uma relação que determina quais delas se conhecem, retorna uma lista de convidados que conhecem no mínimo *N* outros que também irão à festa.

Entrada

A entrada do programa é composta por diversos casos de teste. A primeira linha de cada caso de teste contém três números inteiros *P*, *R* e *N*, separados por espaço, representando, respectivamente, o número de pessoas (variando de 2 a 100), o número de relações entre as pessoas (variando de 1 a *P**(*P*-1)/2) e o número mínimo de convidados que alguém deve conhecer para também ser convidado (variando de 1 a *P*). Cada uma das *R* linhas seguintes contém dois números inteiros (variando de 1 a *P*), separados por espaço, indicando uma relação que determina que as pessoas com os números especificados se conhecem. O último caso de teste é seguido por uma linha que contém três zeros separados por espaço.

Saída

Para cada caso de teste, imprima o número de cada uma das pessoas que serão convidadas, em ordem crescente, uma pessoa por linha.

Exemplo de entrada

```
10 6 2
1 5
3 4
1 4
2 4
3 5
6 1
0 0 0
```

Exemplo de Saída

```
1
3
4
5
```