Corrida

Introdução

Considere o seguinte problema[1].

> *Alice e Beto estão apostando corrida. Ambos correm com a mesma velocidade, andam com a mesma velocidade e correm mais rápido do que andam. A diferença está na resistência deles. Alice corre metade do tempo e anda a outra metade do tempo. Beto corre metade do espaço e anda a outra metade do espaço. Quem ganha a corrida?*

Neste exercício-programa, sua tarefa consiste em fazer um programa que resolve uma versão generalizada desse problema.

Descrição

O comportamento de cada competidor é descrito por uma sequência de intruções. Cada instrução diz se o competidor corre ou anda e por que fração do tempo ou do espaço restante ele faz isso. Tomando como exemplo o problema original enunciado acima, a sequência de Alice teria duas instruções:

> Corre 50% Tempo e Anda 100% Tempo.

Quer dizer que, no início da corrida, Alice corre 50% do tempo que resta para completar a corrida e, feito isso, anda 100% do tempo que resta para completar a corrida a partir daquele ponto. São instruções equivalentes:

> Corre 50% Tempo e Anda 100% Espaço.

No exemplo, a sequência de instruções correspondente ao Beto seria

> Corre 50% Espaço e Anda 100% Espaço.

Para representar as instruções de uma forma que o computador consiga

[1] Adaptação do problema 6 no livro *E aí, algum problema?* de Beth Bürgers e Elis Pacheco, Editora Moderna, São Paulo, 1997.

entender, vamos usar `C` para correr, `A` para andar, `T` para tempo e `E` para espaço. Além disso, vamos substituir porcentagem por dois inteiros representando uma fração positiva menor que 1. Por exemplo, as instruções da Alice ficariam assim.

```
C 1 2 T A 1 1 T
```

Generalizar o problema significa que, agora, podemos ter muitos competidores (não só Alice e Beto) e que cada competidor tem seu comportamento ditado por uma sequência de múltiplas instruções. Por exemplo, poderíamos imaginar uma corrida com três competidores que se comportam de acordo com as regras a seguir.

Competidor 1: `C 1 6 T A 1 4 E C 1 6 E C 1 1 T`
Competidor 2: `C 1 4 T A 2 7 E C 1 1 T`
Competidor 3: `C 3 11 T C 2 7 E A 1 3 T C 1 1 E`

Só para ter certeza de que você entendeu, se o comprimento do percurso fosse 100km, e os competidores corressem a 10km/h e andassem a 5km/h, o Corredor 1 demoraria 12h para completá-lo: correria 20km em 2h (1/6 das 12h), depois andaria 20km (1/4 dos 80km restantes) em 4h, depois correria 10km (1/6 dos 60km restantes) em 1h e, finalmente, correria 50km em 5h (1/1 das 5h restantes). O Corredor 2 empataria com o Corredor 1. Dada essa informação, tente deduzir sozinho quanto tempo e espaço ele corre correspondente a cada uma das instruções. Finalmente, deduza sozinho o tempo que o Corredor 3 demora para completar a corrida e decida se ele é o vencedor ou o perdedor.

Vamos continuar supondo que todos correm com a mesma velocidade, andam com a mesma velocidade e correm mais rápido do que andam. O problema é que, nessa versão mais geral, a relação entre a velocidade de corrida e de caminhada e o comprimento do percurso pode afetar o resultado de quem será o vencedor. Portanto, para que o problema de se determinar o vencedor esteja bem definido, vamos fixar em 10km/h a velocidade de corrida, em 5km/h a velocidade de caminhada e em 100km a distância a ser percorrida.

Este exercício poderá ser feito em duplas, não triplas. O nome dos integrantes deve aparecer em um comentário, no início do arquivo `.c` que será entregue no Tidia. Somente um dos integrantes deve submeter no Tidia.

Entrada e saída

A entrada do seu programa será composta por diversas linhas, cada uma contendo uma sequência de, no máximo, 1000 instruções relativas ao comportamento de um corredor. Cada instrução engloba quatro elementos: um caractere (`C` ou `A`), seguido de dois números inteiros positivos (o segundo maior ou igual que o primeiro), seguidos de um caractere (`T` ou `E`). Os elementos de uma instrução vêm separados uns dos outros por pelo menos um espaço em branco. As instruções também são separadas umas das outras do mesmo modo. Os dois números da última instrução de um corredor serão sempre iguas a 1, denotando que aquele corredor termina o percurso correndo ou andando 100% do percurso restante (tempo ou espaço).

A saída do seu programa será o tempo que cada competidor demora para completar o percurso, impresso com três casas decimais, um número por linha. A i-ésima linha na saída será o tempo correspondente ao o competidor na i-ésima linha da entrada.

Exemplo

Entrada:

```
C 1 2 T A 1 1 T
C 1 2 E A 1 1 E
A 1 3 T C 2 3 E A 1 1 T
A 8 15 T A 1 6 E C 1 1 E
```

Saída correspondente:

```
13.333
15.000
15.000
15.000
```

Leitura

Você já sabe muito bem que, em linguagem C, que as linhas

```
int n;
scanf("%d", &n);
```

[2]Espaço em branco (*white space* em inglês) denota *space*, *tab*, *newline* ou *carriage return*, que são representados em `C` pelos literais `' '`, `'\t'`, `'\n'` e `'\r'`, respectivamente.

ignoram uma quantidade arbitrária de espaços em branco[2] até o primeiro caractere que não seja espaço em branco e, em seguida, tenta ler um número inteiro (`int`).

O problema é que, na hora de ler caracteres (`char`), se fizermos

```
char x;
scanf("%c", &x);
```

o caractere armazenado em `x` será o primeiro que aparecer no *buffer* de leitura, podendo ser um espaço em branco. Para encontrar o primeiro `char` que seja diferente de espaço em branco, podemos fazer manualmente:

```
int r
char x;
do {
  r = scanf("%c", &x);
} while (r == 1 &&
    (x == ' ' || x == '\t' || x == '\n' || x == '\r'));
if (r == 1) {
  // x foi lido com sucesso e x não é espaço em branco
} else {
  // Erro de leitura ou o fim da entrada foi atingido.
}
```

Ou podemomos trocar **todo** o laço `do-while` por

```
scanf(" %c", &x);
```

Note o espaço antes de `%c`! Mas agora fique atento pois, desse modo, o scanf não distingue quando uma linha acaba e começa outra (porque ele engole o `'\n'`). Para este exercício-programa isso não será um problema, pois a última instrução de cada linha sempre tem o numerador e o denominador da fração iguais a 1. De qualquer modo, nunca se esqueça de olhar o valor retornado pela função `scanf` para ver se a leitura foi feita com sucesso.

Para quem quiser ler cada linha da entrada e colocar numa *string* (i.e., num vetor de `char`), e depois processar essa string, uma possibilidade seria ler cada linha usando `getline()` e depois processar a linha usando `sscanf()`. Pesquise como usar essas funções.

Dicas de implementação

Em uma das maneiras de se fazer este exercício programa, você precisará representar uma função linear (i.e. uma função da forma $f(x) = ax + b$,

onde a e b são frações) como uma estrutura e deverá implementar operações binárias de soma e composição, uma operação de produto por escalar e deve ser capaz de inverter uma tal função. A dica é que as instruções devem ser consideradas na ordem inversa para que se possa descobrir o tempo e o espaço percorridos.

Vamos dar um exemplo considerando a entrada:

```
C 2 7 E A 1 3 T C 1 1 E
```

Digamos que antes de executar a última instrução (`C 1 1 E`), ainda resta percorrer S_0 kilometros em t_0 horas. Como este último trecho do percurso será de corrida, então a relação de S_0 e t_0 é

$$S_0 = 10 \cdot t_0.$$

Agora vejamos como seria a situação antes da instrução (`A 1 3 T`). Digamos que, nesse ponto do percurso, ainda resta percorrer S_1 kilometros e t_1 horas **até o final**. Assim, as relações entre t_1 e t_0, e entre S_1 e S_0 são as seguintes:

$$\begin{aligned} t_1 - t_0 &= \frac{1}{3} \cdot t_1, \\ S_1 - S_0 &= 5 \cdot (t_1 - t_0). \end{aligned}$$

Simplificando, temos

$$\begin{aligned} t_1 &= \frac{3}{2} \cdot t_0, \\ S_1 &= 5 \cdot \frac{1}{3} \cdot t_1 + S_0. \end{aligned}$$

Já temos t_1 em função de t_0. Se quisermos S_1 em função de t_0, teremos

$$S_1 = \left(5 \cdot \frac{1}{2} + 10\right) \cdot t_0.$$

Mais um passo. Agora estamos antes da instrução `C 2 7 E` e nos resta percorrer S_2 kilometros e resta t_2 horas. As relações entre t_2 e t_1, e entre S_2 e S_1 são as seguintes:

$$\begin{aligned} S_2 - S_1 &= \frac{2}{7} \cdot S_2, \\ S_2 - S_1 &= 10 \cdot (t_2 - t_1). \end{aligned}$$

Simplificando, temos

$$\begin{aligned} S_2 &= \frac{7}{5} \cdot S_1, \\ t_2 &= \frac{1}{10} \cdot \frac{2}{5} \cdot S_1 + t_1. \end{aligned}$$

Como S_1 e t_1 são funções de t_0, temos

$$S_2 = \frac{7}{5} \cdot \left(5 \cdot \frac{1}{2} + 10\right) \cdot t_0,$$
$$t_2 = \frac{1}{10} \cdot \frac{2}{5} \cdot \left(5 \cdot \frac{1}{2} + 10\right) \cdot t_0 + \frac{3}{2} \cdot t_0.$$

Após simplificar, temos

$$S_2 = 17.5 \cdot t_0,$$
$$t_2 = 2 \cdot t_0.$$

Como conbinamos que o percurso todo terá 100km, $S_2 = 100$. Agora podemos deduzir o valor de t_0 e usá-lo para calcular t_2 que é o que o exercício pede.

$$t_2 = 2 \cdot 100/17.5 = 11.42857142857142...$$

Portanto, a saída correspondente à entrada examinada seria:

```
11.429
```