```
------
? a
-a-a-a
? e
Não ocorre
-a-a-a
? b
ba-a-a
? n
banana
Adivinhou em 4 tentativas
```

Figura 1: Exemplo de interacção do jogo de adivinha.

Programas interativos

5.1 Escreva um programa completo que reproduza a funcionalidade do utilitário `wc` de Unix: ler um ficheiro de texto da entrada-padrão e imprimir o número de linhas, número de palavras e de *bytes*. Exemplo:

```
$ echo "a maria tinha um cordeirinho" | wc
      1       5      29
```

Sugestão: utilize as funções *words* e *lines* do prelúdio-padrão.

5.2 Escreva um programa completo que lê linhas de texto da entrada-padrão e imprime cada linha invertida.

5.3 Escreva um programa completo que codifique a entrada-padrão usando a cifra de César de 13 posições (ver a aula teórica e ainda o sítio `http://www.rot13.com`). Exemplo:

```
$ echo "a maria tinha um cordeirinho" | ./rot13
n znevn gvaun hz pbeqrvevaub
```

5.4 Escreva uma função interactiva *adivinha* :: *String* $\rightarrow$ *IO* () que implemente um jogo de advinha duma palavra secreta dada como argumento pelo utilizador; um outro jogador vai tentar adivinhá-la.

O programa deve mostrar a palavra, substituindo as letras desconhecidas por traços e pedir uma nova letra; todas as ocorrências dessa letra na palavra devem então ser reveladas. O jogo termina quando o jogador adivinha a palavra; o programa deve então imprimir o número de tentativas (ver Figura 1).

5.5 Escreva uma função *elefantes* :: *Int* $\rightarrow$ *IO* () tal que, por exemplo, *elefantes* 5 imprime os seguintes versos:

Se 2 elefantes incomodam muita gente,
3 elefantes incomodam muito mais!
Se 3 elefantes incomodam muita gente,
4 elefantes incomodam muito mais!
Se 4 elefantes incomodam muita gente,
5 elefantes incomodam muito mais!

Sugestão: utilize a função $show :: Show\ a \Rightarrow a \rightarrow String$ para converter um inteiro numa cadeia de caracteres; pode ainda re-utilizar a função $sequence\_ :: [IO\ a] \rightarrow IO\ ()$ para executar uma lista de ações.

5.6 O jogo *Nim* desenrola-se com cinco filas de peças idênticas (representadas por estrelas), cujo estado inicial é o seguinte:

```
1 :  * * * * *
2 :  * * * *
3 :  * * *
4 :  * *
5 :  *
```

Dois jogadores vão alternadamente retirar uma ou mais estrelas de uma das filas; ganha o jogador que remover a última estrela ou grupo de estrelas.

Implemente este jogo um programa em Haskell que pergunte as jogadas de cada jogador e atualize o tabuleiro. Sugestão: represente estado do jogo como uma lista com o número de estrelas em cada fila; o estado inicial será então $[5, 4, 3, 2, 1]$.