1.

```
#include <stdio.h>

#define MAX_N 81 // 1 a mais que 80 questões, para o '\n'.

int main ()
{
    int n, i, n_acertos;
    char gabarito [MAX_N];
    char prova [MAX_N];

    /* Lê as entradas. Estou supondo aqui uma entrada "honesta", então vai scanf
       para tudo! */
    scanf ("%d", &n);
    scanf ("%s", gabarito);
    scanf ("%s", prova);

    /* Testa a prova. É simples, basta contar quantas vezes uma posição está
       igual na prova e no gabarito. */
    n_acertos = 0;
    for (i = 0; i < n; i++)
        if (prova [i] == gabarito [i])
            n_acertos++;

    printf ("%d\n", n_acertos);

    return (0);
}
```

2.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main ()
{
    int valores [N];
    int i;
    int n_valores = N; /* O vetor começa com N valores, depois perde alguns. */
    int pos;

    /* Gera e mostra o vetor. */
    for (i = 0; i < N; i++)
    {
        valores [i] = rand ();
        printf ("%4d %d\n", i, valores [i]);
    }
    printf ("\n");

    /* Agora fica em um loop lendo e removendo valores... */
    printf ("Remove qual? ");
    scanf ("%d", &pos);
    while (n_valores > 0 && pos >= 0 && pos < n_valores)
    {
        /* Desloca todos os valores a partir da posição dada para a esquerda. */
        for (i = pos+1; i < n_valores; i++)
            valores [i-1] = valores [i];
        n_valores--;

        /* Mostra o vetor. */
        for (i = 0; i < n_valores; i++)
             printf ("%4d %d\n", i, valores [i]);
        printf ("\n");

        if (n_valores > 0)
        {
            /* Lê outra posição. */
            printf ("Remove qual? ");
            scanf ("%d", &pos);
        }
    }

    return (0);
}
```

3.

```
#include <string.h>
#include <stdio.h>

/* Função auxiliar, diz se um caractere é uma letra. */
int ehLetra (char c) {
    return ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'));
}

#define OFFSET_R_L ('R' - 'L') /* A distância do R para o L na tabela ASCII. */

void stling (char* s, char* s_cebolinha)
{
    int i;
    int size = strlen (s); /* Assumindo uma string bem formada. */
    int pos_cebolinha = 0; /* A posição do caractere em s_cebolinha. */

    for (i = 0; i < size; i++)
    {
        /* Verifica se a letra é um R ou r e não está no final da palavra.
           Para saber se é o final de uma palavra, checamos se o próximo
           caractere é uma letra. É seguro fazer isso porque mesmo no final da
           string, existe um '\0'. */
        if ((s[i] == 'R' || s[i] == 'r') && ehLetra (s[i+1]))
        {
            /* Se a letra anterior era também um R/r, já colocamos o L/l na
               posição anterior! */
            if ((i == 0) || (s[i-1] != 'R' && s[i-1] != 'r'))
                s_cebolinha [pos_cebolinha++] = s[i] - OFFSET_R_L;
        }
        else
            s_cebolinha [pos_cebolinha++] = s[i];
    }

    /* Finaliza a string. */
    s_cebolinha [pos_cebolinha] = 0;
}

#define BUFLEN 2048

int main ()
{
    char s [BUFLEN];
    char cebolinha [BUFLEN];

    fgets (s, BUFLEN, stdin);
    s [strlen (s)-1] = 0; /* Remove o '\n'. */

    stling (s, cebolinha);
    printf ("%s\n", cebolinha);

    return (0);
}
```

4.

```
#include <stdio.h>
#include <string.h>

#define BUFLEN 1024

/*============================================================================*/
/* Função que diz se a letra é uma vogal. */

int ehVogal (char c)
{
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' ||
            c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');
}

/*============================================================================*/
/* A solução aqui é percorrer a string da direita para a esquerda e da esquerda
   para a direita ao mesmo tempo. É parecido com o problema do palíndromo, mas
   precisamos de contadores independentes para a direita e a esquerda, que só
   andam quando a posição tiver uma vogal. */

int ehRisadaEngracadaV1 (char* risada)
{
    int n, pos_l, pos_r;

    // Acha o tamanho. Supõe uma string bem formada. Poderia usar strlen também.
    for (n = 0; risada [n] != '\0'; n++);

    // Contador de posições nas duas direções.
    pos_l = 0;
    pos_r = n-1;

    // Percorre até achar 2 vogais. Para quando a esquerda passar a direita.
    while (pos_l < pos_r)
    {
        // Acha a próxima vogal pela ESQUERDA.
        while (!ehVogal (risada [pos_l]))
            pos_l++;

        // Acha a próxima vogal pela DIREITA.
        while (!ehVogal (risada [pos_r]))
            pos_r--;

        /* Note que pode pegar ainda uma vez com pos_l >= pos_r, mas é só 1
           teste a mais, então vou deixar passar. Daria para evitar jogando o
           teste para antes dos 2 whiles, mas aí precisaria ter eles fora do
           loop uma primeira vez. */
        if (risada [pos_l] != risada [pos_r])
            return (0); // Se não for risada engraçada, já dá para retornar.

        // Vai para a próxima vogal.
        pos_l++;
        pos_r--;
    }

    return (1);
}
```

```
/*============================================================================*/
/* Uma solução alternativa - menos eficiente mas mais "limpinha" - é criar uma
   versão da string somente com as vogais, e aí usar uma função genérica para
   identificar palíndromos. */

/* Retorna 1 se uma string dada é um palíndromo, ou 0 do contrário. */
int ehPalindromo (char* s)
{
    int i;
    int size = strlen (s); // Supondo que a string é bem formada.

    // Só precisa ir até o meio!
    for (i = 0; i < size/2; i++)
        /* O caractere na posição i deve ser igual àquele na posição size-1-i.
           O -1 é porque o último índice é size-1. */
        if (s [i] != s [size-1-i])
            return (0);

    return (1);
}

/*----------------------------------------------------------------------------*/

int ehRisadaEngracada (char* risada)
{
    char so_vogais [BUFLEN]; // Seria melhor usar alocação dinâmica, mas...
    int i, n;

    // Mantém somente as vogais.
    n = 0;
    for (i = 0; risada [i] != '\0'; i++)
        if (ehVogal (risada [i]))
            so_vogais [n++] = risada [i];
    so_vogais [n] = '\0'; // Lembrando de fechar tudo!

    // Agora dá para usar a função que já tínhamos...
    return (ehPalindromo (so_vogais));
}

/*============================================================================*/

/* Um main simples para testar. */
int main ()
{
    char risada [BUFLEN];

    scanf ("%s", risada); // Nem testei o tamanho...

    printf ("%d\n", ehRisadaEngracada (risada));

    return (0);
}
```