1. [2 pontos] Escreva uma função de protótipo

```
int quadrado(string s);
```

que recebe um string `s` como argumento e devolve `1` caso `s` seja um quadrado e `0` caso contrário. Aqui, dizemos que um string é um quadrado se ele for a concatenação de dois strings iguais. Por exemplo, `abab` é um quadrado, já que ele é a concatenação do string `ab` com ele mesmo (podemos escrever $\texttt{abab} = (\texttt{ab})^2$). O string vazio (de comprimento 0) é a concatenação dele com ele mesmo, de forma que ele é também um quadrado.

Para esta questão, você deve usar (entre outras) a seguinte função de `strlib`.

```
/*
 * Function: SubString
 * Usage: t = SubString(s, p1, p2);
 * --------------------------------
 * SubString returns a copy of the substring of s consisting of the
 * characters between index positions p1 and p2, inclusive.  The
 * following special cases apply:
 *
 * 1. If p1 is less than 0, it is assumed to be 0.
 * 2. If p2 is greater than the index of the last string position,
 *    which is StringLength(s) - 1, then p2 is set equal to
 *    StringLength(s) - 1.
 * 3. If p2 < p1, SubString returns the empty string.
 */
string SubString(string s, int p1, int p2);
```

2. [2 pontos] Considere a seguinte função

```
double p(double x, double a[], int n)
{
  double temp = 0; int k;
  for (k = n; k >= 0; k--) temp = temp * x + a[k];
  return temp;
}
```

(*i*) Suponha que temos um vetor `c[]` com `c[0] = -2`, `c[1] = -1` e `c[2] = 1`. Qual é o valor devolvido pelas chamadas `p(2, c, 2)`, `p(-1, c, 2)` e `p(1, c, 2)`?

(*ii*) Suponha que os elementos de um vetor `c[]` são tais que `c[i] = i + 1` para todo $0 \leq \texttt{i} \leq \texttt{n}$. Encontre uma fórmula para o valor devolvido pela chamada `p(x, c, n)`. [Ignore erros de arredondamento.]

3. [3 pontos] Por simplicidade, nesta questão você deve assumir que `m` e `n` são sempre inteiros não-negativos primos entre si. Procure justificar suas respostas desta questão com especial cuidado.

(*i*) Escreva uma função de protótipo

```
void f(int x, int *a, int *b, int m, int n);
```

que recebe um inteiro `x` e que põe em `*a` o valor `x` mod `m` e põe em `*b` o valor `x` mod `n`.

(*ii*) Suponha agora que você tem à disposição uma função de protótipo

```
void euclides(int m, int n, int *r, int *s);
```

que recebe os inteiros `m` e `n` e põe em `*r` e `*s`, respectivamente, inteiros α e β tais que $\alpha\mathtt{m} + \beta\mathtt{n} = 1$. Escreva uma função de protótipo

```
void g(int m, int n, int *b1, int *b2);
```

que recebe os inteiros `m` e `n` e põe em `*b1` e `*b2`, respectivamente, inteiros e_1 e e_2 tais que

$$e_1 \equiv 1 \pmod{\mathtt{m}}, \qquad e_1 \equiv 0 \pmod{\mathtt{n}}, \qquad e_2 \equiv 0 \pmod{\mathtt{m}}, \qquad e_2 \equiv 1 \pmod{\mathtt{n}}.$$

[*Sugestão.* Considere os números $\alpha\mathtt{m}$ e $\beta\mathtt{n}$.]

(iii) Escreva uma função de protótipo

```
int h(int a, int b, int m, int n);
```

que recebe inteiros `a`, `b`, `m` e `n` e devolve um inteiro x tal que

$$\begin{cases} x \equiv a \pmod{\mathtt{m}} \\ x \equiv b \pmod{\mathtt{n}}. \end{cases}$$

[*Sugestão.* Considere $ae_1 + be_2$.]

4. [3 pontos]

(i) Escreva uma função de protótipo

```
void elimine(int a[], int i, int n);
```

que elimina do vetor fornecido `a[]` o item `a[i]`. Os elementos `a[i+1],...,a[n-1]` devem ser 'deslocados' uma posição para a esquerda, de forma a 'fechar' o 'espaço' deixado pela eliminação de `a[i]`. Você pode assumir que $0 \le \mathtt{i} \le \mathtt{n} - 1$.

(ii) Escreva uma função de protótipo

```
int reduza(int a[], int i, int n);
```

que recebe um vetor `a[]` com `n` entradas e faz o seguinte: `reduza()` verifica se o valor `a[i]` ocorre como `a[j]` para algum `j > i`. Se isto não ocorrer, `reduza()` deixa o vetor `a[]` intacto e devolve o valor `n`. Caso contrário, `reduza()` deve eliminar de `a[]` todas as ocorrências do valor `a[i]`, a menos daquela ocorrência como `a[i]` (isto é, todos os `a[j]` com `a[j] = a[i]` e `j > i` devem ser eliminados). A função `reduza()` deve devolver o número de elementos restantes no vetor, após a eliminação de todas as duplicações do valor `a[i]` como descrito acima. Você deve usar a função de (i), mesmo que você não a tenha escrito.

(iii) Escreva uma função de protótipo

```
int elimine_duplicacoes(int a[], int n);
```

que elimina do vetor fornecido `a[]` todos os valores duplicados, deixando nele uma ocorrência de cada valor distinto que nele ocorre. O usuário desta função passa em `n` o número de elementos do vetor `a[]`; por sua vez, a função `elimine_duplicacoes()` deve devolver o número de elementos restantes no vetor `a[]` após todas as eliminações de duplicações. Você deve usar a função de (ii), mesmo que você não a tenha escrito.