Zadanie 1.

Napisać funkcję `allSeq n`, która generuje wszystkie ciągi binarne o długości n w postaci listy list. Na przykład

```
allSeq 2 = [[0,0],[0,1],[1,0],[1,1]].
```

Kolejność ciągów w wynikowej liście nie ma znaczenia. W rozwiązaniu proszę użyć `map`.

Zadanie 2.

Napisać funkcję `solve n`, która zwraca listę wszystkich trójek (x, y, z) będącymi rozwiązaniami równania

$$x^2 - 2y^2 = z^5$$

takimi, że $x, y, z \in \{1, 2, \ldots, n\}$. W rozwiązaniu należy wykorzystać funkcję `filter` **lub** zakresy.

Jak zmodyfikować to rozwiązanie, aby funkcja ta generowała *tylko parzyste* rozwiązania tego równania?

Zadanie 3.

- Napisać funkcje `divisors n`, która zwraca listę wszystkich dzielników liczby n.
- Korzystając z tej funkcji napisać funkcję `perfects n`, która zwraca wszystkie liczby doskonałe w przedziale $[1..n]$.

Liczba doskonała to taka, która równa jest sumie swoich dzielników właściwych np. 6 jest doskonała, bo $6 = 1 + 2 + 3$.

Zadanie 4 (2 pkt.)

Rozważmy następujący algorytm generowania permutacji zbioru n-elementowego.

1. Permutacją zbioru jednoelementowego jest zbiór złożony z tego zbioru.
2. Wygeneruj rekurencyjnie wszystkie permutacje zbioru $(n-1)$-elementowego.
3. Do każdej permutacji na każdej pozycji wstaw n.

Napisać funkcję `perm n`, która generuje wszystkie permutacje zbioru $[1, 2, \ldots, n]$ w postaci listy list, tzn.

$$perm\ 3 = [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]].$$

Zadanie 5.

Napisać funkcję `prime`, która dla wywołania `prime n` zwraca listę wszystkich liczb pierwszych z przedziału [2..n] wygenerowanych algorytmem sita Eratostenesa.

Zadanie 6.

Napisać funkcję `subsets`, która dla wywołania `subsets n k` generuje (w postaci listy list) wszystkie k-elementowe podzbiory zbioru 1,2,...,n.

Zadanie domowe.

Wiadomo, że pierwiastek kwadratowy z liczby dodatniej a może być przybliżany wyrazami ciągu (x_n) określone rekurencyjnie w następujący sposób:

$$\begin{cases} x_0 = a \\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \text{ dla } n \geqslant 0. \end{cases}$$

Napisać funkcję `squareRoot :: Double -> Double -> Double`, która dla wywołania `squareRoot a e` zwraca pierwiastek z a wyliczony z dokładnością ϵ. W rozwiązaniu należy użyć `iterate`.

Zadanie domowe.

Rozważmy homomorfizm $h : \{a, b\}^* \to \{a, b\}^*$ określony przez warunki $h(a) = ab, h(b) = ba$. Słowem Thuego-Morse'a nazywamy nieskończone słowo powstałe w wyniku (nieskończonej) iteracji h na literze a. Początkowy fragment słowa to np.

$$h(a) = ab,\ h(h(a)) = h(ab) = h(a)h(b) = abba,$$
$$h^3(a) = h(h^2(a)) = h(abba) = abbabaab$$

Napisać funkcję `thueMorse n`, która zwraca najdłuższy prefiks słowa Thuego-Morse'a krótszy niż n.

Na przykład `thueMorse 20 = "abbabaabbaababba"`.