Zadanie 1.

Napisać moduł `lucas`, który eksportuje jedną funkcję `lucas/1` obliczającą $n-$tą liczbę Lucasa. Za pomocą tego programu obliczyć `lucas(30)`.
Liczby Lucasa zdefiniowane są rekurencyjnie w następujący sposób

$$\begin{cases} L_0 = 2 \\ L_1 = 1 \\ L_n = L_{n-1} + L_{n-2} \text{ dla } n > 1 \end{cases}$$

Zadanie 2.

Napisać moduł `list2`, który eksportuje dwie funkcje `even_pos/1` oraz `dlugosc/1`. Pierwsza funkcja powinna zwracać listę elementów, które w liście wejściowej występują na parzystych pozycjach. Listy numerujemy tutaj od zera. Dla przykładu `even_pos([1,2,3,4,5,6,7]) = [1,3,5,7]`.
Druga z nich zwraca po po prostu długość listy.

Zadanie 3.

Napisać moduł eksportujący funkcję `posNeg/1`, która zwraca parę liczb `{a,b}` taką, że *a* oznacza liczbę dodatnich elementów na liście, a *b* liczbę ujemnych elementów.

Zadanie 4.

Napisać moduł `splitter`, który eksportuje funkcję `split/1`.
Funkcja ta dla wywołania `split (Lista)` zwrócić parę dwóch list: `{A,B}`. Pierwsza z nich zawiera (w kolejności) wszystkie liczby całkowite z listy wejściowej, druga resztę elementów. Na przykład: `split([1,"ala",ola,23,zosia,4]) -> {[1,23,4],["ala",ola,zosia]}.`
Do rozpoznania typu danego obiektu służą funkcję `is_integer/1,is_float/1,is_tuple/1,is_list/1,is_atom/1`, które zwracają `true`, jeśli argument jest liczbą całkowitą, zmiennoprzecinkową,krotką itd.

Zadanie 5.

Napisać moduł `rownanie` eksportujący funkcję `rownanie/3`, która powinna dla wywołania `rownanie(A,B,C)` powinna zwracać `brakRozwiazan`, jeśli równanie

$$Ax^2 + Bx + C = 0$$

nie ma rozwiązań rzeczywistych. Jeśli równanie to ma jedno rozwiązanie, to wynikiem tej funkcji powinna być właśnie ta liczba; w przypadku dwóch rozwiązań x_1, x_2 funkcja powinna zwrócić parę `{x1,x2}`. Uwaga: Pierwiastek z x obliczamy za pomocą `math:sqrt(x)`.

Zadanie 6.

W tym zadaniu listy utożsamiamy ze zbiorami. Napisać moduł eksportujący jedną funkcję `subsets/2`, która dla wywołania `subsets(L,K)` wygeneruje listę wszystkich $K-$elementowych podzbiorów zbioru (listy) L.

1. W rozwiązaniu można założyć, że elementy listy L będą parami różne.
2. W rozwiązaniu można także użyć funkcji `lists:map/2`, która działa analogicznie jak Haskellowa funkcja o tej nazwie.