Wszystkie programy powinny wyświetlać swoje wyniki za pomocą *System.out*. Dane wejściowe mogą być pobierane od użytkownika przez *System.in* albo zapisane na stałe w kodzie funkcji main. **Implementacja powinna składać się minimum z jednej funkcji, zawierającej opisaną logikę i z wywołania tej funkcji w funkcji main.** Pobieranie danych od użytkownika i wyświetlenie wyniku musi być w funkcji main.

1. Napisz program, który wyświetli na konsoli ciąg znaków:
   ****
   Java jest super!
   ****
2. Napisz program, który obliczy sześcian zadanej liczby rzeczywistej.
3. Napisz program, który wyświetla nieparzyste liczby naturalne (od 0) aż do górnej granicy, otrzymanej na wejściu.
4. Wyświetlić liczby od 1 do 20. Jeśli liczba jest podzielna przez 3, to zamiast niej musi wyświetlić się “Fizz”. Jeśli liczba jest podzielna przez 5, to zamiast niej musi wyświetlić się “Buzz”. Postaraj się zminimalizować ilość potrzebnych warunków. To popularna zagadka o nazwie “FizzBuzz”.
5. Iloczyn
   a. Wczytaj ilość liczb od użytkownika
   b. Wczytaj liczby od użytkownika
   c. Oblicz iloczyn wszystkich tych liczb
   d. Wyświetl wynik
   e. Bonus: odrzucaj 0 wpisane przez użytkownika, prosząc o ponowne podanie danej liczby
6. Napisz program, który otrzymuje na wejściu ilość liczb. Następnie prosi o podanie tych liczb, zapisuje liczby w tablicy. Następnie oblicza średnią arytmetyczną liczb zapisanych w tej tablicy.
7. Napisz program, który na wejściu otrzyma limit. Będzie dodawać kolejne liczby naturalne (od 0) tak długo, aż suma osiągnie bądź przekroczy dany limit. Wyświetla otrzymaną sumę i ilość dodanych do siebie liczb.
8. Napisz program, który stworzy 100-elementową tablicę i wypełni ją losowymi liczbami z przedziału 0-20, a następnie policzy i wyświetli wystąpienia poszczególnych liczb w tej tablicy.
9. Napisz program, który otrzymuje na wejściu ilość liczb. Następnie prosi o podanie tych liczb, zapisuje liczby w tablicy. Następnie sortuje za pomocą *Arrays.sort()* i pokazuje 50. i 95. percentyl wyników

(odpowiednio: *szukany_procent=0.5* i *szukany_procent=0.95*).
*<indeks_percentylu> = <podłoga>(<maksymalny_indeks_tablicy> * <szukany_procent>)*
Żeby zmienić wynik *Math.floor()* z double na int, potrzebnego do indeksowania tablicy, trzeba użyć rzutowania: *(int)*.
Ostatecznie: *<indeks_percentylu_w_posortowanej_tablicy> = (int)Math.floor(<maksymalny_indeks_tablicy> * <szukany_procent>);*

10. Napisz program, który wylosuje liczbę naturalną, a jako wynik poda ją oraz sumę jej cyfr.
11. PESEL (Źródło: https://en.wikipedia.org/wiki/PESEL)
    a. Wczytaj PESEL użytkownika
    b. Weź pierwsze 10 cyfr numeru
    c. Pomnóż każdą z cyfr przez odpowiednią wagę:
       1, 3, 7, 9, 1, 3, 7, 9, 1, 3
    d. Oblicz sumę tych iloczynów
    e. Oblicz resztę z dzielenia przez 10. Jeśli jest większa od 0, to odejmij ją od 10, aby otrzymać wynik. W przeciwnym wypadku reszta stanowi wynik.
    f. Jeśli wynik jest równy 11. cyfrze numeru PESEL, to numer jest poprawny
    g. Wyświetl informację o poprawności numeru użytkownikowi
12. Napisz program, który będzie obliczać i wyświetlać przybliżoną wartość nieskończonego szeregu w postaci:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

czyli: 1. wyraz ciągu = 1,
2. wyraz ciągu = $1 + \frac{1}{4}$,
3. wyraz ciągu = $1 + \frac{1}{4} + \frac{1}{9}$,
4. wyraz ciągu = $1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16}$ itd.

Zakończ obliczenia, kiedy wartość kolejnego wyrazu szeregu będzie mniejsza od zadanej przez użytkownika dokładności (np. 0,000001).
Żeby zobaczyć tak małe różnice i nie męczyć się z zapisem naukowym, trzeba użyć typu *BigDecimal*, a przy wyświetlaniu metody *toPlainString()*.

$$\lim_{x \to \infty} \sum_{n=1}^{x} \frac{1}{n^2} = \frac{\pi^2}{6}$$

13. Napisz program, który obliczy pierwiastek zadanej nieujemnej liczby rzeczywistej. Porównaj swój wynik z wynikiem działania funkcji

Math.sqrt().

Algorytm Herona do wyznaczania pierwiastka kwadratowego:

a. przyjmij $b_1 = a$.
b. Obliczaj kolejne $b_n = \frac{b_{n-1} + \frac{a}{b_{n-1}}}{2}$, dopóki odległość między b_n i a/b_n nie będzie mniejsza bądź równa zadanej przez użytkownika dokładności (np. 0,000001).
c. Ostatnie obliczone b_n jest przybliżeniem pierwiastka kwadratowego podanej liczby.

14. Napisz program, który będzie zawierać funkcję, która przyjmuje 3 argumenty: **a**, **b**, **c**. Potraktuj argumenty funkcji jako współczynniki trójmianu kwadratowego $ax^2 + bx + c$ i oblicz pierwiastki tego równania. Pierwiastki równania kwadratowego są dane wzorem: $\frac{-b \pm \sqrt{\Delta}}{2a}$, gdzie $\Delta = b^2 - 4ac$. Równanie może mieć 0, 1 albo 2 pierwiastki, zależnie od wartości Δ.

    Sprawdź implementację na przykładach:

    a. $x^2 + x$ -> x=-1 *albo* x=0
    b. $3x^2 - 6x + 3$ -> x=1
    c. $\frac{1}{2}x^2 + 2x - 4$ -> x=-5.4641 *albo* x=1.4641
    d. $x + 1$ -> *błąd*
    e. $x^2 + x + 1$ -> *brak pierwiastków*

15. Napisz program, który obliczy największy wspólny dzielnik (NWD) wg algorytmu Euklidesa:

    a. Żeby obliczyć *NWD(m, n)*, gdzie *m* i *n* to liczby naturalne:
        i. Jeśli *m=0*, to wynikiem jest *n* - zakończ działanie,
        ii. Oblicz *r* jako resztę z dzielenia *n* przez *m*,
        iii. Wynikiem będzie *NWD(r, m)*.

16. Napisz program, który mając daną macierz kwadratową (tablicę dwuwymiarową o takiej samej ilości wierszy i kolumn) obliczy jej transpozycję (zamieni wiersze z kolumnami) w miejscu (nie używając drugiej struktury danych, modyfikując tylko wejściową).

    Przykładowa tablica:

    int macierz[][] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}, {11, 12, 13, 14, 15}, {16, 17, 18, 19, 20}, {21, 22, 23, 24, 25}};