01. Podstawowe typy - zadania

1.1 Dla dowolnego wektora liczb wyznacz:

A. liczbę elementów dodatnich
B. sumę elementów, które po zaokrągleniu są podzielne przez 3 - policz zarówno sumę elementów przed jak i po zaokrągleniu
C. odległość każdego z elementów od liczby 8
D. jego postać znormalizowaną - minimum przechodzi na -1, maks na 1, a pozostałe elementy mają zostać liniowo przeskalowane według poniższego wzoru, gdzie min, max to ekstrema wektora, a new max/min to docelowy zakres - w naszym przypadku -1 oraz 1

$$x' = \frac{x - \min}{\max - \min} \cdot (new_{\max} - new_{\min}) + new_{\min}$$

E. średnią wartość kwadratów liczb większych od 5 lub mniejszych od 2
F. wektor napisów składający się z dwóch wartości: "nieparzysta" oraz "parzysta" w zależności od tego czy dany element jest parzysty lub nie
G. jego średnią (nie używając funkcji mean())
H. jego wariancję (nie używając funkcji var oraz sd)
I. jego minimum i maksimum (nie używając funkcji min i max)

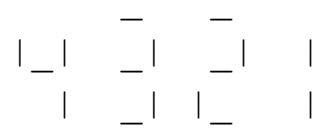
Do testów można użyć losowego wektora liczb całkowitych:
x <- sample(-10:10, size = 10, replace = TRUE)
lub liczb rzeczywistych
x <- round(rnorm(10, 0,1), 2)

1.2 Dany jest wektor liczb całkowitych x o elementach ze zbioru {0,1,...,9} oraz wektory napisów

```
top <- c(" _ ", "  ", " _ ", " _ ", "   ", " _ ", " _ ", " _ ", " _ ", " _ ")
mid <- c("| |", " |", " _|", " _|", "|_|", "|_ ", "|_ ", "  |", "|_|", "|_|")
bot <- c("|_|", " |", "|_ ", " _|", "  |", " _|", "|_|", "  |", "|_|", " _|")
```

Napisz kod, który kolejne cyfry z x wypisze w konsoli w kalkulatorowym stylu. Przykład:
x <- 4:1

...



1.3 Dla danego wektora x, zwróć wektor zawierający tylko unikatowe wartości. Zabronione jest używanie funkcji `unique(), duplicated()` oraz `anyDuplicated()`
Wskazówka 1 - elementy nie muszą zachować pierwotnej kolejności
Wskazówka 2 - użyj funkcji `rle()`

1.4 Dla dwóch wektorów równej długości x i y oblicz ich korelację ze wzoru:

A. $$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

B. $$r = r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum x_i^2 - n\bar{x}^2)}\sqrt{(\sum y_i^2 - n\bar{y}^2)}}.$$

C. $$r = r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}.$$

A następnie przetestuj dla wektorów i porównaj wynik z funkcją cor()

```
x <- 1:10; y <- 10:1
x <- rnorm(20, 0, 1); y <- 5*x+pi
x <- rnorm(20, 0, 1); y <- -2*rev(abs(x))+1
```

1.5 Dla dowolnego wektora x wyznacz indeksy `i` takich elementów, że element `i+1` jest jemu równy.
Przykład - dla wektora c(1,1,0,0,2,5,5) wynik powinien być c(1,3,6)

1.6 Dla wektora o parzystej długości wyznacz wektor o połowę krótszy, którego pierwszy element to suma pierwszego i ostatniego, drugi to suma drugiego i przedostatniego itd.
Przykład - dla wektora 1:6 wynikiem powinien być wektor c(7,7,7)
dla c(1,2,3,3,2,1) wynikiem powinien być c(2,4,6)

1.7 Dla danego wektora liczbowego zawierającego braki danych (`NA`) należy uzupełnić je średnią wartością pozostałych, poprawnych wartości.

1.8 Wypisz w postaci wektora dziesięć pierwszych liczb rozwinięcia dziesiętnego liczby pi (korzystając ze stałej R-owej `pi`)
Wynik powinien być następujący:
```
[1] 3 1 4 1 5 9 2 6 5 3 5
```

1.9 Korzystając ze wzoru Leibniza $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \pi/4$ oblicz przybliżoną wartość liczby pi dla 1 000, 10 000 i 100 000 początkowych wyrazów i porównaj uzyskane liczby z wartością R-owej stałej `pi`.

1.10 Podana wyżej metoda nie jest jedyną na przybliżanie liczby pi. Skorzystamy teraz z metody Monte Carlo, której algorytm wygląda następująco:

1. Wylosuj n punktów z dwuwymiarowej przestrzeni [-1,1] x [-1,1]
2. Sprawdź ile punktów jest oddalonych od punkt (0,0) o mniej niż 1
3. Podziel tę liczbę przez n i przemnóż przez 4

Do losowania punktów użyj funkcji runif (sprawdź w dokumentacji jak)

1.11 Dla danego wektora liczbowego napisz kod, który zwróci listę zawierającą zawsze trzy elementy:

- wektor elementów mniejszych od 0
- wektor elementów równych 0
- wektor elementów większych od 0

Przykład dla wektora -2:2
```
[[1]]
```

```
[1] -2 -1
[[2]]
[1] 0
[[3]]
[1] 1 2
```