Przetwarzanie danych tekstowych (napisów)

Zadanie 9.1 (*MG*).
Napisz funkcję, która generuje wszystkie możliwe kombinacje n pierwszych małych liter alfabetu angielskiego (por. obiekt `letters`) i m kolejnych liczb naturalnych.

```
kombinuj(3, 2)
## [1] "a1" "b1" "c1" "a2" "b2" "c2"
```

Zadanie 9.2 (*MG*).
Złącz kolumny `year`, `month` i `day` ramki danych `nycflights13::weather` w jeden wektor napisów. Zastosuj format dat postaci `RRRR-MM-DD` (dokładnie 10 znaków na napis).

```
print(wynik[c(1, 100, 250, 6789)])
## [1] "2013-01-01" "2013-01-05" "2013-01-11" "2013-10-11"
```

Zadanie 9.3 (*MG*).
Każdy lodowiec z ramki danych `nasaweather::glaciers` ma przyporządkowany 11 lub 12 znakowy unikatowy identyfikator (nadany zgodnie z konwencją WGMS), składający się z pięciu części, kolejno:

- kod terytorialny (2 znaki);
- kod kontynentu (1 znak);
- kod zlewiska (4 znaki);
- kod pozycji (2 znaki);
- kod pozycji szczegółowy (2–3 znaki).

Na podstawie obiektu `glaciers` utwórz ramkę danych o pięciu kolumnach zawierającą powyższe informacje (odseparowane).

```
sample(nasaweather::glaciers$id, 4)
## [1] "PE1D352AC13"  "RB1D22125235" "RB1D22125116" "RB1D22125237"
```

Zadanie 9.4 (*AO; Pełnoletniość*).
Załóżmy, że mamy dany wektor napisów zawierający informacje o datach urodzenia losowo wybranych osób. Dane te zapisane są w formacie *rrrr-mm-dd*. Napisz funkcję, która zwróci ramkę danych o czterech kolumnach: oddzielnie rok, miesiąc i dzień oraz informacja, czy dana osoba jest dziś pełnoletnia, czy nie (na podstawie aktualnej daty systemowej). Dla niepoprawnych dat (np. *2011-02-29*) wstawiaj wartości `NA`.

Zadanie 9.5 (*MG*).
Zastąp w podanym napisie wszystkie ciągi kolejno występujących po sobie białych znaków pojedynczymi spacjami.

Zadanie 9.6 (*MG*).
Wyekstrahuj wszystkie #hasztagi z #DokumentuTekstowego.

Zadanie 9.7 (*MG*).
Stwórz wyrażenie regularne, które dopasowuje się do każdej poprawnej stałej liczbowej, w tym: `12.123`, `-53`, `+1e-9`, `-1.2423e10`, `4.` oraz `.2`.

Zadanie 9.8 (*MG*).
Napisz funkcję `qgram()`, która wyznacza wszystkie unikatowe q-gramy słowne (dla danego $q \geqslant 1$) występujące w danym napisie i podaje ich liczby wystąpień. Na przykład, napis `"zachował się bardzo nieprzyzwoicie"` zawiera następujące 2-gramy (bigramy): `"zachował się"`, `"się bardzo"`, `"bardzo nieprzyzwoicie"`.

```
qgram("Przepraszam! Przepraszam! Przepraszam! Służbowo!", 2) # 2-gramy
## przepraszam przepraszam    przepraszam służbowo
##                        2                       1
```

Zadanie 9.9 (*MG*)**.**
Stwórz wyrażenie regularne, które wykrywa wszystkie poprawne (i żadne inne) syntaktycznie nazwy zmiennych w języku R (por. `?make.names` i s. **??**).

Zadanie 9.10 (*MG*)**.**
Wydobądź listę wszystkich adresów e-mail (bez sprawdzania ich poprawności, każdy adres w osobnym napisie) z danego wektora napisów. Interesują nas tylko adresy pojawiające się w następujących kontekstach: „`href="mailto:`*`adres_email`*`"`” lub „`href='mailto:`*`adres_email`*`'`”.

Zadanie 9.11 (*BT; Wyszukiwanie adresów e-mail*)**.**
Napisz funkcję `findemails()`, która dla podanego wektora napisów `x` reprezentujących kolejne paragrafy tekstu zwróci wektor napisów, którego kolejne elementy zawierać będą wszystkie znalezione poprawne (poszukaj definicji w internecie) adresy email w `x`. Przykład:

```
findemails("Jola Lojalna <jl@google.com>")
## [1] "jl@google.com"
findemails(c("a@b@c", "Send comments to Grzegorz.Urlich@math.sk", "ooo@e13.com, wu@ok.edu"))
## [1] "Grzegorz.Urlich@math.sk" "ooo@e13.com"             "wu@ok.edu"
```

Zadanie 9.12 (*BT; Wyszukiwanie adresów www*)**.**
Napisz funkcję `decomposeurls()`, która dla podanego wektora napisów `x` zwróci ramkę danych o `length(x)` wierszach zawierającą – jako wektory napisów – rozłożone na odpowiednie fragmenty poprawne adresy URL (kolumny: `protokol`, `host`, `zasob`). Jeśli dany napis nie odpowiada poprawnym URL, odpowiadający wiersz ramki danych powinien zawierać braki danych.

```
http://www.wikipedia.com/wiki/URL
|      |                  |
|      |                  ścieżka dostępu
|      |                  do zasobu
|      host (adres serwera)
protokół
```

```
decomposeurls(c("http://gagolewski.rexamine.com/publications/", "ala ma kota",
   "ftp://213.135.44.35"))
##   protokol                   host         zasob
## 1     http gagolewski.rexamine.com publications/
## 2     <NA>                   <NA>          <NA>
## 3      ftp           213.135.44.35
```

Zadanie 9.13 (*MG*)**.**
Napisz własną wersję funkcji `stri_trim()`, która usuwa wszystkie białe znaki z początku lub końca każdego z podanych napisów.

Zadanie 9.14 (*MG*)**.**
Dany jest wykaz ścieżek do plików na dysku (zob. np. `list.files()`). Usuń z niego wszystkie pliki nie zakończone rozszerzeniem „`.R`”.

Zadanie 9.15 (*MG: Zdarzenia*)**.**
Dane są trzy wektory o równych długościach, w których i-te elementy opisują, odpowiednio, godzinę, minutę i sekundę zajścia pewnego zdarzenia. Napisz funkcję, która zwróci

$(n-1)$-elementowy wektor informujący o liczbie sekund, które upłynęły między *kolejnymi* (w czasie; wektory trzeba posortować) zdarzeniami.

Zadanie 9.16 (*MG; Generowanie haseł*)**.**
Trudne do złamania hasło składa się z co najmniej jednej małej oraz jednej wielkiej litery alfabetu łacińskiego, a także co najmniej jednej cyfry i chociaż jednego znaku spoza wymienionych zbiorów.

Napisz funkcję `genpwd()`, która generuje `n` (domyślnie 1) `k`-znakowych (domyślnie 8) losowych, trudnych do złamania haseł. „Inne znaki" dane są w postaci niepustego wektora napisów `dodatkowe` (każdy napis o długości 1), domyślnie równego `c("_", "-")`.

Zadanie 9.17 (*BT; Szyfr Cezara*)**.**
Najprostszy (i jednocześnie niezapewniający bezpieczeństwa) ze znanych sposobów szyfrowania tekstu to tzw. szyfr Cezara z przesunięciem `h`. Przekształca on każdą i-tą literę alfabetu łacińskiego na $(i+$`h`$)$-tą (z ewentualnym „zawijaniem"). Napisz funkcję `cezar()`, która dla danego `h` zaszyfruje każdy napis z danego wektora napisów.

```
cezar(c("abcABCąśćxyzXYZ0123!@$", "Ala ma Ferrari."), 1)
## [1] "bcdBCDąśćyzaYZA0123!@$" "Bmb nb Gfssbsj."
cezar("F bodfokrę aw gwę dcrcpo!", -14) # odszyfruj
## [1] "R naprawdę mi się podoba!"
```

Zadanie 9.18 (*MG; Zliczanie częstości występowania liter*)**.**
Napisz funkcję `zlicz()`, która dla danego napisu zliczy liczbę wystąpień każdej litery z alfabetu łacińskiego (i tylko takiej) i zwróci wynik w postaci wektora liczb całkowitych z ustawionym atrybutem `names` (jaka litera). Wynikowy wektor powinien być posortowany nierosnąco względem częstości występowania każdej z liter.

Zadanie 9.19 (*MG; Palindrom*)**.**
Palindrom to napis, który zapisany wprost i wspak jest identyczny. Napisz funkcję `palindrom()`, która zwraca tylko te napisy z danego wektora napisów, które są palindromami. Np. `palindrom(c("Łał", "bala", "Madam, I'm Adam")) == c("Łał", "Madam, I'm Adam")`. *Uwaga. Przy sprawdzaniu należy pomijać wszystkie znaki niebędące literami i ignorować wielkość liter.*

Zadanie 9.20 (*MG; Sprawdzanie nawiasów*)**.**
Napisz funkcje `sprNawiasy()`, która sprawdza, czy wszystkie nawiasy (różnego rodzaju) w danych napisach są poprawnie pozamykane i zagnieżdżone. Jako argumenty przyjmuje ona:

1. wektor napisów `tekst`;
2. dwukolumnową macierz napisów jednoznakowych `nawiasy`, dla której I kolumna określa nawiasy otwierające (domyślnie `c("(", "{", "[")`), a II – odpowiadające im wersje zamykające (domyślnie `c(")", "}", "]")`).

Funkcja powinna zwracać wektor całkowitoliczbowy o długości równej liczbie napisów w `tekście`. Wartość `0` oznacza, że nawiasy są podane w sposób poprawny. wartość `>0` wskazuje numer znaku, na którym leży pierwszy nawias sprawiający problem.

Wskazówka: nie wystarczy oczywiście sprawdzić, czy liczba nawiasów otwierających jest taka sama, jak zamykających. Dla przykładu, poprawne są napisy `"(a (( b ) ( c )) ( d ) e )"` (nie-nawiasy możemy spokojnie ignorować) oraz `"z <- x[(y>0)]; sum(x)"`. Niepoprawne jest z kolei `"(()"` oraz `"[(])"`.

Jeśli nie potrafisz obsłużyć różnych typów nawiasów, stwórz tylko wersję szczegółową dla jednowierszowej macierzy `nawiasy`.