Homework 3 (String processing)

Select and solve **5** tasks from this list.
Include all the solutions in **one** R script file, see `homework3_template.R` for a template.
When you're done, send them via courses.ipipan.edu.pl.
All the scripts will be examined by plagiarism detection software.

Exercise 03.01. Write a function `eventdifftime()` which expects on input a data frame with n rows (for some n) and the following columns:

- `h` – an integer vector with elements in $\{0, 1, \ldots, 23\}$,
- `m` – an integer vector with elements in $\{0, 1, \ldots, 59\}$,
- `s` – an integer vector with elements in $\{0, 1, \ldots, 59\}$.

Each row represents some event's time (all of the events occur in the same day).

Firstly, the data frame should be sorted w.r.t the events' times (increasingly). The function should return a numeric vector of length $n - 1$, giving the time difference (in seconds) between consecutive events. For example, if the input is like:

```
   h m s
1 13 1 1
2 14 3 4
3 13 2 2
```

then the expected result is $(61, 3662)$.

Exercise 03.02. Write a function `uncomment()` to remove comments from a given source code chunk. Parameters:

1. a character vector `text` – each string represents a single line of a source file, cf. `readLines()`,
2. a 2-column character matrix `tags` – each row either determines how to start and end a comment or how to denote a comment that terminates at the end of the current line (in such a case the value in the 2nd column is `NA`).

For example, in R we have `tags=matrix(c("#", NA), ncol=2)`. On the other hand, in C++ we have `tags=matrix(c("/*", "//", "*/", NA), ncol=2)`, and in PHP – `tags=matrix(c("/*", "//", "#", "*/", NA, NA), ncol=2)`.

The function should return a character vector. If the comment removal results in a blank line, do not include it in the result. By the way, do note that „`1 /* foo bar /* foo // bar */ 2`" should be transformed to „`1 2`".

Exercise 03.03. Write a function `findemails()` to extract all well-formed email addresses from each string in a given character vector. The function should return a character vector. For example:

```
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"
## [3] "wu@ok.edu"
```

HUMAN CAPITAL
NATIONAL COHESION STRATEGY
EUROPEAN UNION
EUROPEAN
SOCIAL FUND
Project co-financed by the European Union under the European Social Fund

Exercise 03.04. Write a function `scrabble()` with the following parameters:

1. `x` – a named integer vector giving how may points (values) may be obtained by using corresponding letters (names),
2. a single integer `n`,
3. a character vector `dict`, determining a set of possible words.

The function should return top (w.r.t. the total number of points per word) `n` words from `dict`. Total number of points for a given word is just the sum of the points per each letter.

For example, `x` may be of the following form:

```
## a ą b c ć d e ę f g h i j k l ł m n ń o ó p r s ś t u w y z ź ż
## 1 5 3 2 6 2 1 5 5 3 3 1 3 2 2 3 2 1 7 1 5 2 1 1 5 2 3 1 2 1 9 5
```

If you'd like some more challenge, submit a second solution that addresses a more complicated problem. You are also given a vector `y` that tells you how many letters are available (e.g. 3 "a"s, 2 "b"s, etc.; originally we assumed that each letter may be re-used infinitely many times). Return top `n` (or less) words that can be formed with a given set of letters.

Exercise 03.05. Write a function `checkParens()` to determine if all opening parentheses in a given text chunk are properly nested. Parameters:

1. a character vector `text`;
2. a 2-column matrix `parens` consisting of single-character strings; the 1st column gives the opening parentheses, and the 2nd one – their closing counterparts, e.g. `c("(", "{", "[")` and `c(")", "}", "]")`.

The function should return an integer vector of the same length as `text`. The value of `0` informs us that the corresponding string consists in properly nested parentheses. Otherwise, a value `>0` denotes the position at which we may find a problematic parenthesis.

For example, `"(a (( b ) ( c )) ( d ) e )"` and `"z <- x[(y>0)]; sum(x)"` have all their parentheses properly nested. On the other hand `"(()"` should result in 1, and `"[(])"` in 3 (or 2, it depends on your implementation).

Exercise 03.06.** You are given a character vector, each string represent a paragraph of text. Write a function `wrap_dynamic()` that implements a dynamic ("minimum raggedness"[1]) word wrap algorithm. Output a character vector such that, when printed on string, it fills up to `h` (76 by default) text columns.

For example:

```
library("stringi")
h <- 55
text <- c("Tatooooooo nieeeeeee wraaaaaaaca; ranki i wieeeeeeeczory
We łzaaaaaach goooooooo czeeeeeeeeeeeeeekam", stri_dup("a ", 55))
cat(wrap_dynamic(text, h), sep=stri_join("\n|", stri_dup("-",h-2), "|\n"))
## Tatooooooo nieeeeeee wraaaaaaaca;
## ranki i wieeeeeeeczory We łzaaaaaach
## goooooooo czeeeeeeeeeeeeeekam
## |-----------------------------------------------------|
## a a a a a a a a a a a a a a a a a a a a a a a a a a a
## a a a a a a a a a a a a a a a a a a a a a a a a a a a
```

[1] See en.wikipedia.org/wiki/Word_wrap.