Perl TD2 : Associative arrays (or hashes)

Summary on associative arrays (or hashes)

Associative arrays (or hashes), like standard arrays, are collections of data, but the indices identifying the elements are not necessarily integers, but any values that are called "keys". The order of couples (key/element) in a hash is unpredictable. It depends only on the way Perl stores them to facilitate its search based on keys. In addition, access to a non-existent hash implies its creation. The name of a hash variable begins with % (Rq : the variables %hash, @hash and $hash have no connection between them).

```
# Assign an element (create the hash if it does not exist)  my %hash;
$hach{"aaa"} = 3; # key = "aaa", element = 3
$hach{5.6} = "abc"; # key = "5.6", element = "abc"
# Global initialization or copy of hash
%hach = ("aaa",3,5.6,"abc"); # as above
#or
%hach = (aaa => 3, 5.6 => abc); # as above
%cop = %hach; # copy of hash
```

Function **keys** : provides a list of all the keys of a hash. Used with a for loop, it allows you to browse all the elements of the hash (without of course controlling the order) :

```
@liste = keys(%hach); # @liste = ("aaa", 5.6) ou (5.6, "aaa")
for $cle (keys(%hach)) {
    print "cle: $cle, valeur: $hach{$cle}\n";
}
```

Function **values** : provides a list of all elements of a hash, in the same order as the list of keys returned by keys :

```
@liste = values(%hach); # @liste = (3,"abc") ou ("abc",3)
```

Function **each** : provides a pair (key/value) in the form of a two-element list. For each application on the same hash, the following pair is returned (empty list if it is finished) :

```
while ( ($cle,$val)=each(%hach) ) {
    print "cle: $cle, valeur: $val\n";
}
```

Function **delete** : Allows you to delete an element from a hash :

```
delete $hach{"aaa"}; # %hach no longer contains the pair cle/element: "aaa"/3
```

Regular expressions.

Regular expressions allow you to manipulate a text in a very powerful way. In particular, they are used to test whether a string contains a certain pattern.

To find a substring :
$m/\$exprreg/$: search operator (match). The 'm' is optional.
The search applies to $_ by default otherwise you must use the operator `=~`

$/\$exprreg/i$: with option i, the search is not case sensitive. The'.' replaces any character, the '^' refers to the beginning of the chain, the'$' refers to the end of the chain.
In a regular expression, you can use the following shortcuts :

- `\d` : a number (equivalent to `[0-9]`)
- `\D` : anything but a number (equivalent to `[^0-9]`))
- `\w` : an alphanumeric character (equivalent to `[0-9a-zA-Z_]`)
- `\W` : everything but an alphanumeric character (equivalent to `[^0-9a-zA-Z\_])`
- `\s` : a blank, equivalent to `[ \n\t\r\f]`
- `\S` : everything but a blank, equivalent to `[\^ \n\t\r\f]`

Perl affects 3 special variables : $&, $‘ et $', which contain respectively the part of the chain that match, the part that is before and the part that is after. Thus the matching position is $length(\$`) + 1$.

To retrieve some parts of the chain that match sub-expressions of the pattern, we places these sub-expressions in parentheses. The different results are stored in variables `$1`, `$2`, ...

Example :

```
if ($time =  /(\d\d):(\d\d):(\d\d)/) { # time in hh:mm:ss format
   $hours = $1;
   $minutes = $2;
   $seconds = $3;
}
```

1 Exercice 1

The following table specifies the disease associated with each gene :

AD4	Alzheimer's Disease
BRCA1	Breast Cancer
DMD	Duchenne Muscular Dystrophy
FMR1	Fragile X Syndrome
GBA	Gaucher's Disease

Write a Perl script that creates a hash whose keys are the elements in the first column and whose values are the elements in the second column. The script will then ask the user to enter a gene name, and it will display the corresponding disease. The program will have to display an error message if the gene entered is not found in the table.
Note : To define the hash, you can use these 2 syntaxes :

```
%genes = ("AD4", "Alzheimer's disease", "BRCA1",
          "Breast Cancer",...);
```

OR

```
%genes = (AD4 => "Alzheimer's disease",
          BRCA1 => Breast Cancer,
          ...);
```

2 Exercice 2

The file **kinases_map.txt** was retrieved from the bank OMIM, it contains a table of information on human protein kinase genes. Each line contains information about a gene, the different fields are separated by a vertical bar. The description of the fields is :

— Gene symbol(s),
— Gene name,
— date of entry to the database,
— Cytogenetic location (chromosome number followed by cytogenetic band(s)),
— Accession number in OMIM.

Write a Perl script that builds a hash whose pairs key/value are the data 'Accession number in OMIM'/'Gene name'found in the file.

3 Exercice 3

Write a perl script that asks the user to enter a string and tests if it contains :

1. the word **kinase** (singular or plural),
2. the pattern **ATAT** repeated at least once,
3. the pattern **CGCG** repeated twice, followed by three symbols, different from **A** and **C**,
4. the pattern **ACG** followed by 3 characters among **A**, **C**, **G** and **T**, followed by **T** repeated between 4 and 6 times.

4 Exercice 4

Consider a file containing on each line the following information, separated by the character **:**

— student number made up of eight digits,
— a name in capital letters,
— a first name (simple or compound) in lower case, except the initial(s), and may include accents,
— a code made up of capital letters and numbers.

Write a perl script that builds a hash table whose keys are student numbers and values are email addresses of students. The email address is made up of the first name while tiny and unaccented (with **-** if it's a first name compound), followed by a period, followed by the name in lower case, followed by the string **@u-bordeaux.fr**.
The script will display the list of students' email addresses.

5 Exercice 5

Consider a file containing any text.

1. Write a Perl script that counts the total number of occurrences of each word in the text. If the text is "les coccinelles dévorent les pucerons", you must get the result :

```
2  les
1  coccinelles
1  dévorent
1  pucerons
```

Hints : Use the operator `each`.

2. Display the total number of words in the text, as well as the number of different words (i.e. each word must be counted only once).
   For the previous example :

   ```
   Total number of worlds: 5
   Number of differents worlds: 4
   ```

   indice : Use the operators `keys` and `values`.

3. Find another solution that does not use the values operator.