Course 'Imperative Programming' (IPC031)
Assignment 11: *Recursion*

1. Background
In this assignment you implement a number of *recursive* functions.

2. Learning objectives
After doing this assignment you are able:
- to implement simple recursive algorithms by a recursive function;
- to reason about recursive functions;
- to realize that complexity is a property of an algorithm and not of a problem (there can be many different algorithms solving the same problem).

3. Assignment
Part 1: The power function
Part 1.1: Naïve power
Implement the recursive equation of the *power* function as a recursive function in C++ (parameter *n* must be a non-negative integer, parameter *x* can be an *integer* or a *double*). Test your code for different, representative values (hint: check the conditions in the recursive equation).

$$power\ (x,n) = \begin{cases} 1 & \textbf{if } n = 0 \\ x \cdot power\ (x,n-1) & \textbf{if } n > 0 \end{cases}$$

Part 1.2: Power, more efficiently
The above realization of the *power* function is naïve in the sense that the computation of *power(x,n)* requires *n* multiplications of *x* and the use of intermediate results. The order of run-time complexity is $\mathcal{O}(n)$. By making use of the property $x^{2n} = x^n \cdot x^n$, or equivalently, $x^{2n} = (x^n)^2$ you can implement a more efficient version. Implement this more efficient version *recursively*. What is the order of run-time complexity of this more efficient algorithm?

Part 2: Palindromes
A *palindrome* is a text that is identical to its reversed version. For instance, `"otto"` and `"lepel"` are palindromes. In this part you develop functions that determine whether a string is a (variant of a) palindrome.

Part 2.1: Straight palindromes
Develop the recursive function:

```
bool palindrome1 (string text, int i, int j)
```

which decides whether `text[i]` … `text[j]` is a palindrome.

Examples:
- `palindrome1 ("otto", 0, 3)`
  returns `true`.
- `palindrome1 ("Otto", 0, 3)`
  returns `false` because `'O'` is not equal to `'o'`.
- `palindrome1 ("Madam, I'm Adam.", 0, 15)`
  returns `false` because `'M'` is not equal to`'.'`.

Part 2.2: Case-insensitive palindromes
Develop the recursive function:

```
bool palindrome2 (string text, int i, int j)
```

which decides whether `text[i]` … `text[j]` is a palindrome, but this time it should consider `'a'` also equal to `'A'`, `'b'` also equal to `'B'`, and so on for all letter characters.
Examples:

- `palindrome2 ("otto", 0, 3)`
  returns `true`.
- `palindrome2 ("Otto", 0, 3)`
  returns `true` because `'O'` is now also considered equal to `'o'`.
- `palindrome2 ("Madam, I'm Adam.", 0, 15)`
  returns `false` because `'M'` is not equal to`'.'`.

Part 2.3: Case-and-space-insensitive palindromes
Develop the recursive function:

```
bool palindrome3 (string text, int i, int j)
```

which decides whether `text[i]` … `text[j]` is a palindrome, but this time it should consider `'a'` also equal to `'A'`, `'b'` also equal to `'B'`, and so on for all letter characters. Moreover, it should ignore all space characters (`' '`) and punctuation marks (`'.'`, `','`, `':'`, `';'`, `'\''`, `'!'`, `'?'`, `'-'`).
Examples:

- `palindrome3 ("otto", 0, 3)`
  returns `true`.
- `palindrome3 ("Otto", 0, 3)`
  returns `true` because `'O'` is now also considered equal to `'o'`.
- `palindrome3 ("Madam, I'm Adam.", 0, 15)`
  returns `true` because case, space, and punctuation marks are ignored (`'.'` at the end), and `'M'` is now also considered equal to `'m'`.

Part 3: Matching characters in a string
Develop the recursive function:

```
bool match_chars (string chars, int i, string source, int j)
```

which decides whether the characters `chars[i]` … `chars[chars.length()-1]` occur in `source[j]` … `source[source.length()-1]` in that order, but allowing to skip characters in `source`.
Examples:

- `match_chars ("abc", 0, "It is a bag of cards", 0)`
  returns `true` because all characters in `"abc"` occur in order: `"It is a bag of cards"`.
- `match_chars ("abc", 0, "It is a bag of books", 0)`
  returns `false` because character `'c'` does not occur: `"It is a bag of books"`.
- `match_chars ("abc", 0, "It is a classy bag", 0)`
  returns `false` because character `'c'` does not occur after `'b'`: `"It is a classy bag"`.

Note that you can readily test the implementation by making the function `match` of *assignment 8: Music database and queries* call `match_chars`, using the actual parameter 0 for `i` and `j`.

4. Products

As product-to-deliver you only need to upload to Brightspace *"main.cpp"* that you have created with solutions for each part of the assignment.

⇒ **Deadline:** *Thursday, November 29, 2018, 13:30h.*