02393 C++ Programming Exercises

Assignment 9

To be handed in via CodeJudge — `https://dtu.codejudge.net/02393-e21/assignments`

The goal of the following assignments is to familiarize with recursion. All the exercises can be solved without recursion but you should try to use recursion. In all cases input is to be read from `cin` (until the end of input is reached) and the result must be provided on `cout`.

1 Reverse

Write a program that reverses a sequence of integers, as provided in the standard input. For example, if the input is

```
1 2 3 4 5
```

then the output should be

```
5 4 3 2 1
```

2 Fibonacci

Write a program that reads from `cin` a sequence of whitespace-separated numbers (until the end of the input), and for each number n, prints the Fibonacci number[1] F_n defined as:

$F_0 = 1 \quad F_1 = 1 \quad F_2 = F_1 + F_0 = 2 \quad F_3 = F_2 + F_1 = 3 \quad F_4 = F_3 + F_2 = 5 \quad F_5 = F_4 + F_3 = 8 \quad \ldots$

For example, if the input to the program is

```
0 5
```

then the program output should be the following (since $F_0 = 1$ and $F_5 = 8$):

```
1 8
```

3 Palindrome

Write a program that reads from `cin` a sequence of whitespace-separated numbers (until the end of the input) and decides whether the sequence is a palindrome, i.e., whether reading the sequence from right to left results in the very same sequence. For example, if the input is

```
13 22 33 22 13
```

then the output should be

```
yes
```

Instead, if the input is

```
13 22 31
```

then the output should be the following (since the sequence read right-to-left is `31 22 13`):

```
no
```

Note. The right-to-left reading does not refer to individual digits, but to entire numbers.

[1] See `https://en.wikipedia.org/wiki/Fibonacci_number`. **Note**: here we use $F_0 = 1$ (a common variation is $F_0 = 0$).

4 The Levenshtein distance

The Levenshtein distance between two sequences of characters $u = u_1, u_2, \ldots, u_k$ and $v = v_1, v_2, \ldots, v_l$ is defined by:

$$\mathrm{d}(u,v) = \begin{cases} |v| & \text{if } |u| = 0 \\ |u| & \text{if } |v| = 0 \\ \min \begin{cases} \mathrm{d}(u^1, v) + 1 \\ \mathrm{d}(u, v^1) + 1 \\ \mathrm{d}(u^1, v^1) + f(u_1, v_1) \end{cases} & \text{otherwise} \end{cases}$$

where:

- $|w|$ denotes the length of a sequence w;
- w_1 denotes the first element of a sequence $w = w_1, w_2, w_3, \ldots$;
- w^1 denotes the suffix $w_2, w_3, \ldots$ of a sequence $w = w_1, w_2, w_3, \ldots$.
  That is, w^1 corresponds to w without the first element (which is w_1);
- and, for e and e' being two characters, $f(e, e')$ is 0 when e and e' are the same character (i.e. $e = e'$), and 1 otherwise.

Write a program that reads two words and outputs their Levenshtein distance. For example, if the program input is:

```
AB B
```

then the program output should be:

```
1
```

This is because (according to the formula above) we have $d(\text{“}AB\text{”}, \text{“}B\text{”}) = 1$. In fact:

$d(\text{“}AB\text{”}, \text{“}B\text{”}) = min(d(\text{“}B\text{”}, \text{“}B\text{”}) + 1, d(\text{“}AB\text{”}, \text{“”}) + 1, d(\text{“}B\text{”}, \text{“”}) + 1) = min(1, 3, 2) = 1$
$d(\text{“}B\text{”}, \text{“}B\text{”}) = min(d(\text{“”}, \text{“}B\text{”}) + 1, d(\text{“}B\text{”}, \text{“”}) + 1, d(\text{“”}, \text{“”}) + 0) = min(2, 2, 0) = 0$
$d(\text{“}AB\text{”}, \text{“”}) = 2$
$d(\text{“}B\text{”}, \text{“”}) = 1$
$d(\text{“”}, \text{“}B\text{”}) = 1$

Hint. STL `string` objects provide a method called `substr` that may be handy...

5 Additional challenges (optional)

- Write a function that, given a set of elements, computes all the possible permutations of the elements.
- Write a function that, given a set, computes its powerset.