02393 C++ Programming Exercises

Assignment 9

Hand-in via `https://dtu.codejudge.net/02393-e17/assignment/`

The goal of the following exercises is for you to familiarize with recursion. All the exercises can be solved without recursion, but we suggest that you try to use recursion. In all cases input is to be read from `cin` and the result is to be provided to `cout`.

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
```

Fibonacci Write a program that computes the Fibonacci numbers F_{n_i} for a sequence $n_0, \ldots n_k$ given in the standard input. See `http://en.wikipedia.org/wiki/Fibonacci_number`

For example if the input is

```
0 5
```

then the output should be

```
1 8
```

since F_0 is 1 and F_5 is 8. No! $F_0 = 0$, $F_5 = 5$

Palindrome Write a program that decides whether a sequence of integers is a palindrome, i.e. if 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
```

but if the input is

```
13 22 31
```

then the output should be

```
no
```

since the right-to-left reading is

```
31 22 13
```

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

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 suffix $w_2, w_3, \ldots$ of a sequence $w = w_1, w_2, w_3, \ldots$; w_1 denotes the first element of a sequence $w = w_1, w_2, w_3, \ldots$; and $f(e, e')$ is 0 when $e = e'$ and 1 otherwise.

As an example you can easily check that the distance $d(\text{“}AB\text{”}, \text{“}B\text{”})$ between “AB” and “B” is 1 since:

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

Write a program that reads two words and returns their distance.

Challenges. Try to write a function that, given a set of elements, computes all the possible permutations of the elements. Try to write a function that, given a set, computes its powerset.