**Preamble.** 1) You need loops at some point. You may use the following:

```perl
while ( CONDITION ) { DO_SOMETHING };
```

E.g.

```perl
while ( @array ) { my $element = shift @array; print $element; }
```

prints out an array. You can of course use for-loops as well. You may even use 'map' to construct loops ...

2) To read lines from standard input, use the following command:

```perl
my @input = <>;
```

This puts the user input into @input, line by line. The input is terminated with Control-D (end of file).

**Exercise 1:** Write a program that reads integer numbers line by line and prints the square of all the even numbers greater than 10.

**Exercise 2:** Write a program that asks ten questions, reads the answers and shows the number of answers that were right. The questions should be like:

Please enter n things

in which n is an arbitrary number from 1 to 3 and and things is one of the three words dollars, stars or commas. The questions should be chosen arbitrarily by the program.

**Exercise 3:** Write a program that reads lines from standard input until end-of-file, then prints the number of lines in the input, followed by the input in reverse order.

Extra Credit: Count the number of characters and words, too. (Only a solution using regular expressions will be given in the solution sheet, but maybe you find a solution using only the language constructs introduced?)

**Exercise 4:** Write a program that recognizes palindromes (words or phrases that read the same in both directions).
Exercise 5: ** If you’d like to analyze a multiple alignment\(^1\) of biosequences like

\[
\begin{align*}
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{---AG-CATATGCTTGGCGGGAAGA} \\
\text{---AG-TATATTCTTTTCTCAAGA} \\
\text{---AG-CATATGCTTGGCGGGAAGA} \\
\text{---TAG-CATGTGCTTGGCGTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATGTGCTTGGCGTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA} \\
\text{CCTGGTGTGATCCTGCCAGTAG-CATATGCTTGTCTCAAGA}
\end{align*}
\]

it is sometimes useful to identify “runs”, i.e. consecutive columns that have the same feature. In the example alignment, columns 1-18 share the feature that there are “-“-characters in rows 6-9. Or, the last four columns share the feature that they are identical. Starting with a list of column indices as the input, we would like to “flag” (identify) the indices involved in runs in a flexible way, so you may think of the following set of parameters:

- \(\delta\), the maximum index difference that defines “consecutive”,
- \(\beta\), the minimum number of consecutive indices of a “long run” which is flagged from start to end,
- \(\alpha\), the number of consecutive indices that are not flagged at the start of a “short run” of length smaller than \(\beta\).

Examples, for \(\delta = 2\), \(\beta = 15\), \(\alpha = 4\):

Type a set of columns, separated by space: 2 4 5 6 7 8 9 10 15 16
Artifacts are: 7, 8, 9, 10
(The run goes from 2 to 10; it is not interrupted by the missing index “3”, since \(\delta = 2\). The run is “short” since it involves 9 indices instead of 15 or more. Since \(\alpha\) equals 4, the run does not involve the first four indices 2,4,5,6.)

Type a set of columns, separated by space: 2 4 5 8 9 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 40
Artifacts are: 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
(There are two short runs from 2 to 9, since 5 and 8 are not consecutive. No index is flagged because \(\alpha = 4\). The run from 15 to 31 involves more than 15 indices, it is a long run for which all indices are flagged.)
