C++ Exercise 1: Periodic Strings

Deadline: Friday, September 4, 2020 by 23:59 EEST Timezone.
How to submit: Please send your GitHub repository (e.g. `workspace-cpp`); create a single subdirectory in this repository named `ex01-periodic`. Place all your project files there (see section "Implementation details" below).
Grading: This exercise is worth 20‰(or 2%) of the total grade.

In a Land Far, Far Away there was a copyright agency named AKKAAKKA. Their talented lawyers found a legal loophole that gave them the right to license the use of certain words and numbers. Every time somebody wrote a book, a poem or a statistical report, it was sent to the copyright agency. The agency charged the author of this text for every use of a word or a number that is periodic. A word or number is called *periodic*, if it can be obtained by concatenating two or more identical fragments.
The agency AKKAAKKA charged the author of a text an integer number of euros, where the sum depends on the length of the shortest period. For example `FF` was charged 1 EUR (as it has 1-letter period), `HOHOHOHO` was charged 2 EUR (as we do not consider the longer period of 4 letters `HOHO`), but the use of non-periodic words were not charged at all.

Your task is to develop a software that receives an input file containing one word (or integer) per line, and for each one shows the shortest period. Or number 0, if no period was found.
On the very first line of the input file we specify the input mode.

- If the mode is `str`, then we search for the shortest period that covers the entire string two or more times.
- If the mode is `dec`, then we search for the shortest period that covers the entire decimal number two or more times (after the possible initial zeros are dropped).
- If the mode is `hex`, then we search for the shortest period that covers the entire hexadecimal number.

Input
The first line has the mode (`str`, `dec` or `hex`). It is immediately followed by newline character (`\n`). After that there are one or more strings (or, respectively, decimal numbers or hexadecimal numbers) that will be analyzed by your program. All input numbers may contain leading zeroes (all the zeroes are ignored when testing the periodicity). All hexadecimal numbers can contain lowercase digits `a,b,c,d,e,f` as well as uppercase digits `A,B,C,D,E,F` (all of them are converted to lowercase when testing the periodicity).
Output
The output should contain the same lines as input (with the exception of the first line – no mode should be written). Every input string (or number, or hexadecimal number) is written to the output, followed by a space and by the length of the shortest period (or 0, if there was no period at all).
Leading zeroes (`dec` or `hex` mode only) should not be displayed; all the hexadecimal digits should be lowercase.
Limitations

- Strings are one per line (up to 1000 long); printable ASCII chars only.
- Integers ("dec" or "hex" mode) are between 0 and $2^{64} - 1 = 18446744073709551615$.
- Input files contain no more than 1000 lines each.

Sample input **test01in.txt**:

```
str
ABCABC
ABCD
ABABA
```

Expected output **test01expected.txt**:

```
ABCABC 3
ABCD 0
ABABA 0
```

Sample input **test02in.txt**:

```
dec
017017
20192019
```

Expected output **test02expected.txt**:

```
17017 0
20192019 4
```

Sample input **test03in.txt**:

```
hex
0012e12E
12F12
```

Expected output **test03expected.txt**:

```
12e12e 3
12f12 0
```

Implementation Details
Your code should contain 3 source files: `Periodic.cpp` (containing an empty constructor and 3 functions), its header file: `Periodic.h`, and the class with main function that processes the I/O `PeriodicMain.cpp`. Here are the functions in `Periodic.cpp`:

```
Periodic();           // an empty constructor
int findPeriod(std::string str);
int findPeriod(int n);
int findPeriodHex(int n); // treat argument as a Hex number
```

During the grading we will run the following script (in Jenkins or similar):

```
# check out the code
git clone <your repository>
# switch to the directory with all the source files
cd ex01-periodic
# Here we remove all your files EXEPT 3 source files specified above.
# Also replace your Makefile with our own.
make all
./ex01periodic < test01in.txt > test01out.txt
diff -B test01out.txt test01expected.txt
# ... more tests - including private tests
```

- Our grading process may also invoke the `findPeriod(...)` functions from your `Periodic.cpp` directly, bypassing the `PeriodicMain.cpp` caller.
- We will run all your projects on the same grading machine; the fastest programs will be specifically mentioned and honored. (Otherwise there are no runtime requirements for this exercise, as long as any input file with 1000 lines can be processed in under 10 seconds.)
- The source of inspiration for this exercise: `https://bit.ly/31TqSZR`, the idea about *data polymorphism* (the same program eats both strings and integers in two different formats), processing them in two identically named functions.