Exercises for Phase 1

For all of the task below, try to handle as much as possible errors which could happen.

1. **Quadratic equation**

Task: write a script which finds all real solutions of quadratic equation following form:

$$A\mathbf{x}^2 + B\mathbf{x} + C = 0$$

Input: script takes 3 space separated (\s*) arguments (A,B,C from above). If less than 3 argument passed - assume 0 for absent arguments.

Output: solutions of quadratic equation should be written in STDOUT, separated by 1 space.
All errors (no real solutions, bad input) should be written in STDERR.
Last line of output should be terminated with \n.
Script should return 0 in case of successful execution, and non-zero result in case of error.

Example of running:

```
quadratic_equation.pl   1 2 3
```

2. **Prime numbers**

Task: given an Integer number. Script should write in STDOUT all Prime numbers, less than given number.

Input: script should ask user to insert integer number *n* from diapason 2..1000000.
Output: space separated prime numbers from 1 till *n*.

3. **Replace all**

Task: given a line of text, string to match and string to replace all matches with.

Input: All strings should be read from STDIN. Script should ask user to input data.
First line of input - text which should be parsed.
Next inputs - space separated string to match and string to replace with. An empty line interrupts entering data by customer and script starts execution. Strings which have been already replaced should not be affected by next replacements.
For example, if I want to change aA to z and zZ to a, input should be like following:

Example:

Execution:

./replace.pl

Standard Input:

```
replace: Please, enter 1 line of text:
Sample text aAzZ
replace: Please, enter space separated strings to match and replace (one pair per line).
replace: Empty line will interrupt input and start execution:
aA zZ
zZ a
text story
------ empty line -----
replace: Sample story zZa
```

BE CAREFUL: From the example above, after first replacement (aA -> zZ) original text will be transformed to Sample text zZzZ. As zZ in red is result of first replacement, it should not be affected by zZ -> a. As a result we should have Sample story zZa.

Output: Refer to example above. Each line of output should be written in STDOUT and last line should be terminated with \n. Errors should be written in STDERR

4. **Sort numerically**

Given a sequence of numbers in English, one per line, e.g.:

one hundred thirty three
two thousand one
five

… output them by their equivalent numeric value, from largest to smallest, e.g.:

two thousand one
one hundred thirty three
five

Each number will be less than one million.

Input: name of the file with numbers as an argument.
Output: print sorted numbers in STDOUT one per line.

5. **The shortest path**

Given a network of directed connections, output the minimum of steps between a named pair of nodes. The first line of the input will be a pair of node names which are the source and the target:

nodeA nodeB

All subsequent lines starting from the 2nd line will just be a list of single links between pairs of nodes. Together, these define the connectivity of the network. You should print the minimum of steps required to get from nodeA to nodeB given the definition of the network, e.g.:

node83 node16
node83 node12
node12 node11
node83 node01
node12 node04
node04 node05
node04 node16
node12 node16

Input: name of the data file as an argument.
Output: You should print 2 in STDOUT as the goal is to get from node83 to node16 (first line defines source and destination nodes).
Print in STDOUT -1 if there is no solution.

6. **Figure renderer**

Is required to create a program that draws geometric figures and calculates their area, based on coordinate points.

Type of supported figures:

- Rectangle
- Square
- Circle
- Equilateral triangle

The allowed commands and params will be:

- create
    - figure type
        - Rectangle

        - 4 coordinate points
    - Square
        - 4 coordinate points
    - Circle
        - Center
        - One point in the perimeter
    - Triangle
        - 3 Points

Requirements:

- OOP
- Class diagram
- The result should be the draw and area of the created figure.
- Each figure type should be draw with different colors
- A coordinate point is composed by an X,Y dupla.

Propose an application that fits to this need.

7. Databases

Once we have the program from task 5 working that process an input and write the results in the STDOUT.
Now we need to store each created figure in a MySql database. You need to install mysql and create the analog data model from our domain model. Then add create simple queries to store the data

Hints:

https://metacpan.org/pod/DBI DBI documentation.

Add list commands

To conclude this exercise what we need is to add an extra command that will list the figures for a specific type.

- list
    - figure type

The output should be a list with all the figures stored in the DB for that particular type