In each exercise make your source code and output readable.

Exercise 1. Write a program that computes the sum of the digits of a nonnegative integer. Integer should be given by the user, do not accept values less than 0. For example, the sum of digits of 1234 is 1+2+3+4 = 10.

Exercise 2. Write a program that reads a positive at most 5-digit integer, and creates a new integer by adding one more digit at the end. The last digit so-called checksum is the remainder in division by 10 of the sum of the digits of the integer given by the user. For example from 123 we should get 1236, from 88 we should get 886.

Exercise 3. Write a program that reads a positive at most 6-digit integer and then checks whether all digits are equal.

Exercise 4. Write a program that reads a positive at most 6-digit integer and then checks whether all digits are even.

Exercise 5. Write a program that reads a positive at least 2-digit and at most 6-digit integer and then checks whether the last digit is a correct checksum which is the remainder in division by 10 of the sum of the digits except for the last digit. Integer should be given by the user, do not accept values less than 10 or greater than 999999. For example: correct numbers are: 1236, 886, 9997, 33; incorrect numbers are: 2227, 12, 888, 9999

Exercise 6. Write a program that reads a positive integer N, and outputs the information how many positive consecutive integers should be added to obtain the smallest number greater than N. For example, the smallest number greater than 10 is 15 and is obtained by the addition of 5 consecutive positive integers.

Exercise 7. Write a program that computes the greatest common divisor and the smallest common multiple of two positive integers.

Exercises 8-11 are a bit complex.

Exercise 8. Since numerology assumes that numbers are of great importance for each person, it might be good to know how to calculate your own number.
The easiest way is to identify your number based on your date of birth. We can do it by adding each digit from day, month and year separately. We add digits until we get the result from 1 to 9. The only exception to this rule is the so-called master numbers, consisting of two identical digits (thus: 11, 22, 33 etc.) - in this case we do not sum them up, but we leave them in this double form.
For example:
16.05.1998 = 1 + 6 + 5 + 1 + 9 + 9 + 8 = 39 next 3 + 9 = 12 next 1+2=3
However; we can add first of 16+5+1998=2019 next 2+0+1+9=12 and next 1+2=3
For 15.05.1998 we get 15+5+1998=2018 next 2+0+1+8=11

Exercise 9. Write a program that gets from the user positive integers up to getting non-positive number, and checks whether the integer is prime and moreover outputs all prime factors.
Interaction with the program might look like this:

```
Enter positive number 34
34 = 1 * 2 * 17
Enter positive number 123456
123456 = 1 * 2 * 2 * 2 * 2 * 2 * 2 * 3 * 643
Enter positive number 643
643 = 1 * 643 (prime number)
```

Exercise 10. Write a program that reads a positive integer, and outputs a new integer such that each digit from original integer is incremented, e.g. from 1598 one should get 26109.

Exercise 11. Write a program that reads a positive integer, and outputs a new integer such that to the digit of unity 1 is added, to the digit of tens 2 is added, to the digit of hundreds 3 is added, to the digit of thousands 4 is added and e.t.c. , e.g. from 1598 one should get 58119