1/3rd term Programming Fundamentals

Oct. 3 2025, 18:30-21:30h

- You can solve the problems in any order. Solutions must be submitted to the automated judgement system Themis. For each problem, Themis will test ten different inputs, and check whether the outputs are correct.
- Grading: You get one grade point for free. The remaining nine points are based solely on the judgement given by Themis. The first problem is worth one grade point. The remaining four problems are worth two grade points each.
- You can score partial points if a program passes several but not all test cases. For example, if you pass 6 out of 10 test cases, then you are awarded 60% of the grade points for that exercise.
- Inefficient programs may be rejected by Themis. In such cases, the error will be 'time limit exceeded'. The time limit for each problem is one second.
- The number of submissions to Themis is not limited. No points are subtracted for multiple submissions.
- There will be no assessment of programming style. However, accepted solutions are checked manually for cheating: For example, precomputed answers will not be accepted, even though Themis accepts them.
- Needless to say, you are not allowed to work together. If plagiarism is detected, both parties (supplier of the code and the person that sends in copied code) will be excluded from any further participation in the course.
- You are not allowed to use email, phones, smartwatches, tablets, calculators, etc. There is a calculator available on the exam computers (see icon on the desktop). You are allowed to consult the ANSI C book and a dictionary. You are not allowed to use a printed copy of the reader or the lecture slides, however they are available digitally (as a pdf) in Themis. You are allowed to access your own submissions previously made to Themis.
- For each problem, the first three test cases (input files) are available on Themis. These input files, and the corresponding output files, are called `1.in` , `2.in`, `3.in`, `1.out`, `2.out` and `3.out`. These files can be used to test whether the output of your program matches the requested layout, so that there can be no misunderstanding about the layout and spaces in the output.
- **If you fail to pass a problem for a specific test case, then you are advised not to lose much time on debugging your program, and continue with another problem. In the last hour of the exam, all test cases will be made visible in Themis.**

Problem 1: BMI

The **Body Mass Index (BMI)** is a widely used measure to assess whether a person has a healthy body weight for their height. It is calculated using the formula

$$\text{BMI} = \frac{\text{weight in kilograms}}{(\text{height in meters})^2}.$$

Although the BMI does not directly measure body fat, it provides a simple numerical value that can be compared against standard categories to indicate whether a person is `UNDERWEIGHT`, at a `HEALTHY` weight, `OVERWEIGHT`, or `OBESE`.

This exercise asks you to write a program that reads a person's weight (in kg, a positive `int`) and height (in cm, a positive `int`), and outputs the corresponding category (see table below).

BMI Range	**Category**
BMI < 18.5	`UNDERWEIGHT`
$18.5 \leq$ BMI < 25.0	`HEALTHY`
$25.0 \leq$ BMI < 30.0	`OVERWEIGHT`
BMI ≥ 30.0	`OBESE`

Example 1:
input:
```
70 170
```
output:
```
HEALTHY
```

Example 2:
input:
```
85 170
```
output:
```
OVERWEIGHT
```

Example 3:
input:
```
100 180
```
output:
```
OBESE
```

Example 4:
input:
```
50 180
```
output:
```
UNDERWEIGHT
```

Problem 2: Summing products until stability

A positive integer n can be transformed iteratively in the following way: If the number has more than one digit, form a new number by taking every pair of neighbouring digits, multiplying the two digits in each pair, and adding all those products together. The result of this addition becomes the new value of n for the next iteration. If the number has only one digit, it is considered *stable* and the process stops.
For example, beginning with **2357** the process evolves as follows:

$$2357 \longrightarrow \underbrace{2\times3+3\times5+5\times7}_{6+15+35=56} \longrightarrow \underbrace{5\times6}_{30} \longrightarrow \underbrace{3\times0}_{0}.$$

The final single digit is 0, reached after 3 transformation steps.

The input is a positive integer n. The program should output two numbers, separated by a space. The first number is the single digit obtained after repeatedly applying the described transformation to n. The second number is the number of steps required to reach that digit.

Example 1:
input:
```
2357
```
output:
```
0 3
```

Example 2:
input:
```
234214
```
output:
```
6 2
```

Example 3:
input:
```
987654
```
output:
```
4 2
```

Problem 3: Pisano numbers

As you know, the Fibonacci sequence is defined by

$$F_0 = 0, \qquad F_1 = 1, \qquad F_k = F_{k-1} + F_{k-2} \text{ for } k \geq 2.$$

This yields the following infinite series of numbers:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, \ldots$$

When we take the Fibonacci sequence modulo some integer $m \geq 1$, the sequence of remainders eventually begins to repeat. This repeating sequence is called the *Pisano sequence*, and its length is called the *Pisano period* for m. For any value of $m > 1$, the repetition always starts again with the pair $(0, 1)$.

For example, if $m = 7$, then the Fibonacci sequence modulo 7 is

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 1,\ 6,\ 0,\ 6,\ 6,\ 5,\ 4,\ 2,\ 6,\ 1,\ 0,\ 1, \ldots$$

Here the pattern starts repeating after 16 numbers, because $F_{16} = 987 = 7 \times 141$ and $F_{17} = 1597 = 7 \times 228 + 1$. So, the Pisano period for $m = 7$ is 16.

The input is an integer m, where $1 \leq m \leq 100000 = 10^5$. The output must be the Pisano period for m.

Example 1:	Example 2:	Example 3:
input:	**input**:	**input**:
`7`	`10`	`13`
output:	**output**:	**output**:
`16`	`60`	`28`

Problem 4: Palindromic rotation

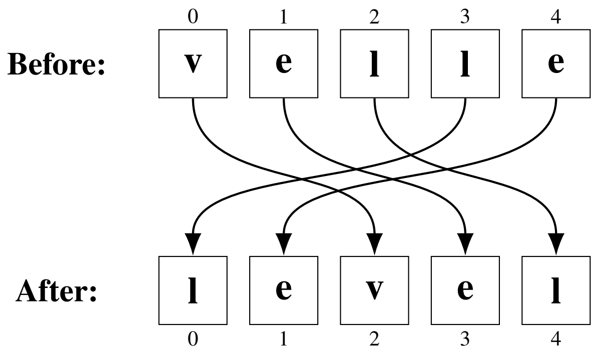
Recall that a *palindrome* is a word that reads exactly the same from left to right as it does from right to left.

A *cyclic left rotation* moves every letter of a word one place to the left, taking the first letter and appending it to the end.

A k*-cyclic left rotation* is obtained by applying this single-step left rotation k times, so each letter is shifted k places to the left while the first k letters wrap around.

Starting from the word `velle`, a 3-cyclic left rotation moves each letter three places to the left, bringing the first three letters to the end. The resulting word is `level`, as you can see in the following figure. Clearly, `level` is a palindrome because it reads the same forward and backward.

Cyclic left rotation by 3: `velle` → `level`



The input is a single word consisting only of lowercase letters from the English alphabet. You may assume that the word contains at most 20 letters. The output must be the *k-cyclic left rotation* of this word (with minimal k) that is a palindrome. If such a rotation does not exist then the output must be `###`.

Example 1:	Example 2:	Example 3:
input:	**input**:	**input**:
`velle`	`abab`	`damma`
output:	**output**:	**output**:
`level`	`###`	`madam`

Problem 5: Descriptive numbers

You are given a starting integer n with $0 \leq n < 100$. We define an iterative process on the number as follows: Read the digits from left to right and *describe* consecutive runs of identical digits by writing first the run length (in decimal) and then the digit itself. The result is the next number.

Example. Starting from $n = 42$:

$$42 \rightarrow 1412 \rightarrow 11141112 \rightarrow 31143112 \rightarrow \cdots$$

Explanation:

- 42: "one 4, one 2" $\Rightarrow$ 1412.
- 1412: "one 1, one 4, one 1, one 2" $\Rightarrow$ 11141112.
- 11141112: "three 1s, one 4, three 1s, one 2" $\Rightarrow$ 31143112.

The input for this problem consists of two numbers n (where $0 \leq n < 100$) and s (where $0 \leq s \leq 20$). The output must be the number that is obtained by applying the above transformation exactly s times, starting from the decimal representation of n.

Note that the output can become quite long. For the given input bounds, the largest output length occurs for $n = 30$ at $s = 20$, which produces a number of 964 digits.

Example 1:

input:

```
42 3
```

output:

```
31143112
```

Example 2:

input:

```
42 1
```

output:

```
1412
```

Example 3:

input:

```
0 5
```

output:

```
1113122110
```