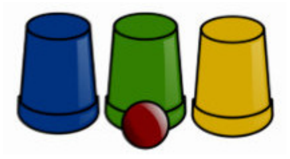
Resit mid-exam Imperative Programming
Oct. 13 2017, 14:00-17:00h

- 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 checks whether the outputs are correct.
- Grading: you get one grade point for free. The remaining nine points are based solely on the judgment given by Themis. The first problem is worth one grade point. The remaining four problems are worth two grade points each, of which you score the full two points if you passed the complete test set of the problem (i.e. 10 test cases), or one grade point if you passed at least 5 (out of 10) test cases.
- 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 unlimited. 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.
- Note the hints that Themis gives when your program fails a test.
- 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 a copy) will be excluded from any further participation in the course. You are not allowed to use email, phones, tablets, calculators, etc. There is a calculator available on the exam computers (see icon on the desktop). You are allowed to consult the reader, the book, handouts of the lecture slides, a dictionary, and submissions previously made to Themis.
- For each problem, the first three test cases (input files) are available on Themis. These examples are the first three test cases of a test set. 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 midterm, all input files will be made visible in Themis (not the output files).**

Problem 1: Shell game

In the *shell game* (in the Dutch language this game is named '*balletje-balletje*), three cups are placed face-down on a surface. A small ball is placed beneath one of these cups so that it cannot be seen. Next, they are then shuffled by the operator in plain view. After several shuffles, a spectator may guess under which cup the ball resides.



We number the positions of the cups (from left to right) 1, 2, and 3. In the above figure, the ball is placed under the cup that is at position 2. Next, a series of pair-wise shuffles is performed. Your program should print the position of the ball after these shuffles.
The input of this problem consists of a line containing the initial location of the ball. The remaining lines contain pairs of locations that are swapped. The series is terminated by a zero.

Example 1:

input:

```
1
1 3
2 1
3 2
2 1
3 1
3 2
1 2
1 3
0
```

output:

```
POSITION 3
```

Example 2:

input:

```
1
3 1
3 1
3 1
3 1
3 1
1 2
2 1
2 1
2 1
0
```

output:

```
POSITION 3
```

Example 3:

input:

```
1
2 3
3 1
2 3
1 2
1 2
1 3
3 1
0
```

output:

```
POSITION 2
```

Problem 2: Friday the 13th

Today, it is Friday the 13th of October 2017. The next year in which the 13th of October is again a Friday is 2023. Recall that a year has 365 days, except for *leap years* which have 366 days. A year is a leap year if it is divisible by 4, but not by 100. Exceptions are years that are divisible by 400 (these are leap years).
Write a program that reads from the input a positive integer `n` (where $n < 1000$), and outputs the `n`th next year (starting from 2017 for n=0) in which the 13th of October is a Friday.

Example 1:

input:

```
0
```

output:

```
2017
```

Example 2:

input:

```
1
```

output:

```
2023
```

Example 3:

input:

```
2
```

output:

```
2028
```

Problem 3: ABC

In this problem we consider strings of characters containing only the letters `'a'`, `'b'` and `'c'`. A *prefix* of a string S is any non-empty leading contiguous part of S. Note that S is a prefix of itself. For example, if $S =$ `"abccba"` then its prefixes are:

`"a", "ab", "abc", "abcc", "abccb", "abccba"`

In this problem, a string S is called *valid* if it satifies the following rules:

- For any prefix of S, the number of `b`'s should not exceed the number of `a`'s.
- For any prefix of S, the number of `c`'s should not exceed the total number of `a`'s and `b`'s.

For example, the string `"abccba"` is *invalid*, because its prefix `"abccb"` contains more `b`'s than `a`'s. The string `aabcccca"` is also invalid because the prefix `"aabcccc"` contains more `c`'s (4) than `a`'s and `b`'s (3). The string `"abcc"` is clearly valid.

Write a program that accepts on the input a string that is terminated by a full stop (`'.'`), and outputs whether the input strings is valid or invalid.

Example 1:
input:
```
abcc.
```
output:
```
VALID
```

Example 2:
input:
```
abccba.
```
output:
```
INVALID
```

Example 3:
input:
```
aabcccca.
```
output:
```
INVALID
```

Problem 4: Encapsulating primes

A *prime* is an integer greater than 1 that has no positive divisors other than 1 and itself. We call a prime number $n > 100$ an encapsulating prime if the number that is obtained by removing its first and last digits is also a prime. For example, the number 14797 is an encapsulating prime because the number 479 is also a prime. Note that we ignore leading zeroes after removal of the first digit. For example, 20897 is an encapsulating prime because 089 = 89 is a prime. Write a program that accepts an integer `n` from the input (where $100 \leq n \leq 1000000000 = 10^9$), and outputs whether the input number is an encapsulating prime or not.

Example 1:
input:
```
14797
```
output:
```
YES
```

Example 2:
input:
```
20897
```
output:
```
YES
```

Example 3:
input:
```
1427
```
output:
```
NO
```

Problem 5: Remainders

Consider the following set of equations in which `%` denotes the modulus-operator (i.e. remainder of integer division):

`x%5 = 1` and `x%3 = 1` and `x%2 = 0`

The smallest non-negative integer `x` that satisifes all three equations is 16.

The input for this problem consists of three lines. Each lines consists of two numbers: a remainder and the corresponding modulus. Write a program that reads the input, and prints the smallest non-negative integer that satisfies the three equations.

Example 1:

input:

```
1 5
1 3
0 2
```

output:

```
16
```

Example 2:

input:

```
1 9
4 5
1 2
```

output:

```
19
```

Example 3:

input:

```
4 7
1 2
6 9
```

output:

```
123
```