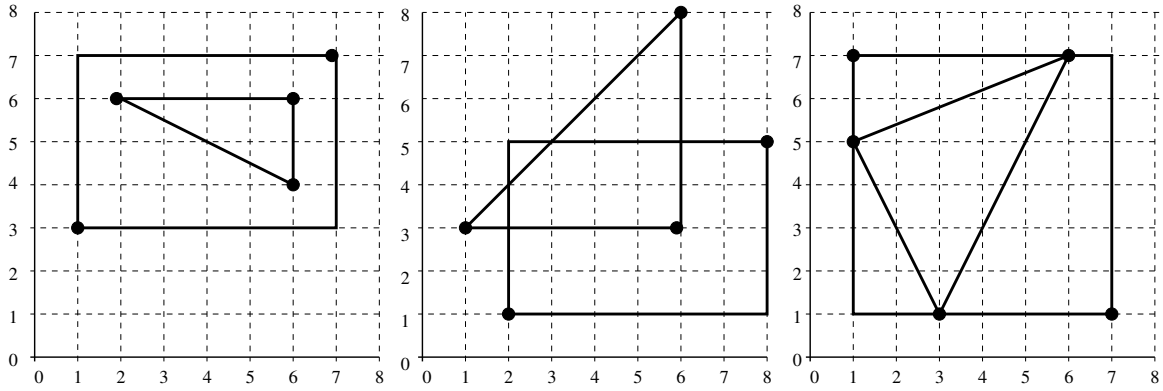
Mid-exam Imperative Programming
Sept. 28 2018, 14:00-17:00h

- 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 copied code) 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 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 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: Enclosed Triangle

The input of this problem consists of integer pairs denoting grid points. The first two pairs are the coordinates of two corner points of the diagonal of an axis aligned rectangle. Next follow three grid points, that define a triangle. Your program should output `YES` if the triangle lies completely inside the rectangle, otherwise it should output `NO`. Note that a point that is located on a side of the rectangle is considered to be 'inside' the rectangle.

The following three figures correspond with the given three sample inputs.



Example 1:

input:

```
1 3
7 7
2 6
6 4
6 6
```

output:

```
YES
```

Example 2:

input:

```
8 5
2 1
6 8
1 3
6 3
```

output:

```
NO
```

Example 3:

input:

```
1 7
7 1
3 1
1 5
6 7
```

output:

```
YES
```

Problem 2: Double Palindromes

A *decimal palindromic number* is a number that remains the same when its digits in decimal notation are reversed. A *binary palindromic number* is a number that remains the same when its bits in binary notation are reversed (ignoring leading zeroes). We call a number a *double palindrome* if it is both a decimal palindromic number and a binary palindromic number. An example of such a double palindrome is 313, which is clearly a palindrome in decimal in notation. Its binary representation is 100111001 ($= 1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$), which is also a palindrome.

Write a program that reads from the input two positive integers a and b (where $0 \leq a \leq b \leq 10^7$), and outputs the number of double palindromes n with $a \leq n \leq b$.

Example 1:

input:

```
313 313
```

output:

```
1
```

Example 2:

input:

```
100 1000
```

output:

```
3
```

Example 3:

input:

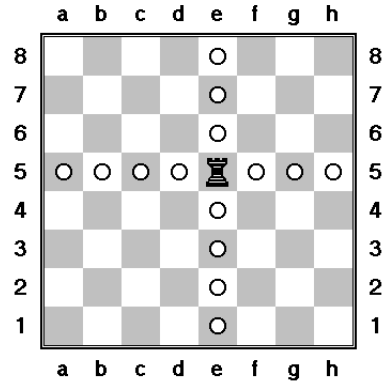
```
1000 1000000
```

output:

```
9
```

Problem 3: Rook Moves

A chess board has 64 squares, labeled a1 (lower left corner) upto h8 (upper right corner). A rook may move any number of steps horizontally or vertically, as can be seen in the following figure. The rook at e5 may move vertically to e1, e2, e3, e4, e6, e7, or e8, while it may move horizontally to a5, b5, c5, d5, f5, g5, or h5.



The input for this problem consists of a line containing the initial position of a rook. The remaining input is a series of moves represented by pairs of integers, terminated by the pair `0 0`. A move consists of a pair of integers, of which the first number is the change of location in horizontal direction (negative for a move to the left, and positive for a move to the right). The second number is the change of location in vertical direction (negative for a downwards move, and positive for an upwards move). The output of the program should be the final location of the rook after applying the series of moves, or `INVALID` if at least one of the moves is invalid (i.e. not a valid rook move, or the rook is moved oustide the boundaries of the board).

Example 1:

input:
```
e5
0 3
0 0
```
output:
```
e8
```

Example 2:

input:
```
e5
-4 0
0 -3
0 0
```
output:
```
a2
```

Example 3:

input:
```
e5
3 3
0 1
0 0
```
output:
```
INVALID
```

Problem 4: Primal Primes

A *prime* is an integer greater than 1 that has no positive divisors other than 1 and itself. The first 10 primes are:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29$$

If we number the primes with an index that starts from 1, then the prime 2 has index 1, the prime 3 has index 2, 5 has the index 3, 7 has index 4, 11 has index 5, and so on.

A prime is called a *primal prime* if its index is also a prime. Of the above primes, the numbers 3, 5, 11, and 17 are primal primes, because their respective indexes 2, 3, 5, and 7 are prime.

Write a program that reads from the input an integer n ($1 \leq n \leq 10000$), and outputs the nth primal prime.

Example 1:

input:
```
1
```
output:
```
3
```

Example 2:

input:
```
2
```
output:
```
5
```

Example 3:

input:
```
4
```
output:
```
17
```

Problem 5: Pythagorean Triples

A *Pythagorean triple* (a, b, c) consists of three positive integers a, b, and c, such that $a < b$ and $a^2 + b^2 = c^2$. The name is derived from the Pythagorean theorem, stating that every right triangle has side lengths satisfying $a^2 + b^2 = c^2$. Clearly, if we scale a right triangle with side lengths a, b, and c by an integer $k > 1$, we find the Pythagorean triples $(k \cdot a, k \cdot b, k \cdot c)$. We call such triples *derived triples*. A Pythagorean triple which is not a derived triple is called a *primitive triple*.

A well-known primitive triple is $(3, 4, 5)$ since $3^2 + 4^2 = 5^2$. Two examples of its derived triples are $(6, 8, 10)$ and $(9, 12, 15)$.

Write a program that reads from the input a positive integer n, and outputs the number of primitive Pythagorean triples (a, b, c) such that $a + b + c = n$. You may assume that $1 \leq n \leq 30000 = 3 \times 10^4$.

Example 1:

input:
```
12
```
output:
```
1
```

Example 2:

input:
```
42
```
output:
```
0
```

Example 3:

input:
```
14280
```
output:
```
3
```