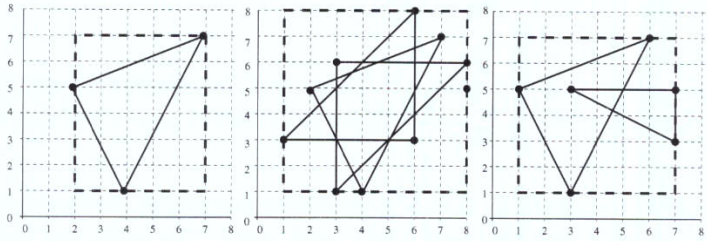
Resit Mid-exam Imperative Programming
Oct. 12 2018, 14:00-17:00h

- 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 check 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 your own 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: Minimum Bounding Rectangle

The input of this problem consists of a series of triangles, and the output should be the lower-left and upper-right coordinates of the axis-aligned *minimum bounding rectangle* of these triangles. The first input line contains a positive integer n, the number of triangles. The remaining n lines contain three grid points (in the order x_0 y_0 x_1 y_1 x_2 y_2) that define a triangle. Note that a point that is located on a side of the bounding rectangle is considered to be 'inside' the bounding rectangle.

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



Example 1:

input:

```
1
4 1 7 7 2 5
```

output:

```
2 1 7 7
```

Example 2:

input:

```
3
1 3 6 3 6 8
7 7 2 5 4 1
3 6 8 6 3 1
```

output:

```
1 1 8 8
```

Example 3:

input:

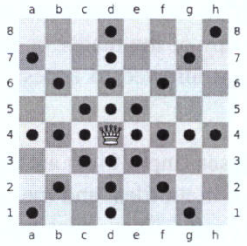
```
2
1 5 6 7 3 1
3 5 7 5 7 3
```

output:

```
1 1 7 7
```

Problem 2: Queen Moves

A chess board has 64 squares, labeled a1 (lower left corner) up to h8 (upper right corner). A queen can be moved to any number of unoccupied squares in a straight line vertically, horizontally, or diagonally as can be seen in the following figure. The queen at d4 may move vertically to d1, d2, d3, d5, d6, d7, or d8. She may move horizontally to a4, b4, c4, e4, f4, g4, or h4. She may move diagonally to a1, b2, c3, e5, f6, g7, h8, a7, b6, c5, e3, f2, or g1. So, in this example she can be moved to 27 valid locations.



The input for this problem consists of a line containing the position of a queen on a chessboard. The output of the program must be the number of valid squares that the queen can move to.

Example 1:

input:

```
d4
```

output:

```
27
```

Example 2:

input:

```
a1
```

output:

```
21
```

Example 3:

input:

```
g7
```

output:

```
23
```

Problem 3: Takuzu Numbers

A positive integer is called a *takuzu* number if its binary representation satisfies the following two rules:

- Ignoring leading 0-bits, the number of 1-bits is equal to the number of 0-bits.
- More than two adjacent bits cannot have the same value.

For example, the number $42(= 1\cdot 2^5+0\cdot 2^4+1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0)$ has the binary representation 101010, which satisfies the above rules so 42 is a takuzu number. The number $21(= 1\cdot 2^4+0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0)$ has the binary representation 10101, which viloates the first rule (three 1-bits, and two 0-bits). The number, $197(= 1\cdot 2^7+1\cdot 2^6+0\cdot 2^5+0\cdot 2^4+0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0)$ has the binary representation 11000101 which satsifies the first rule, but it violates the second rule (three adjacent 0-bits). Hence, the numbers 21 and 197 are not takuzu numbers. The first 20 takuzu numbers are:

$$2, 9, 10, 12, 37, 38, 41, 42, 44, 50, 52, 147, 149, 150, 153, 154, 165, 166, 169, 170$$

Write a program that reads from the input a positive integers n, and outputs the nth takuzu number.

Example 1:

input:
```
1
```
output:
```
2
```

Example 2:

input:
```
8
```
output:
```
42
```

Example 3:

input:
```
20
```
output:
```
170
```

Problem 4: PDS numbers

Let n be a positive integer, and $s(n)$ the function that replaces each digit d of n by the remainder of division of d^2 by 10. For example $s(12345) = 14965$. If we apply the function s again, we find $s(14965) = 16165$. Repeating this again, we find that $s(16165) = 16165$ and notice that we reached a loop and stop repeating this process. It is not very hard to see that for any number n, this repetition will end.

We call a number a PDS (Primal Digit Square) number if all numbers that are reached in the above described repetition are primes. An example of a PDS number is 234317, since 234317, $s(234317) = 496919$, and $s(496919) = 616111$ are all prime, and $s(616111) = 6161111$.
Write a program that reads from the input two positive integers a and b ($1 \leq a < b \leq 1000000 = 10^6$), and outputs the number of PDS numbers n such that $a \leq n \leq b$.

Example 1:

input:
```
1 100
```
output:
```
10
```

Example 2:

input:
```
1 1000
```
output:
```
34
```

Example 3:

input:
```
219300 250000
```
output:
```
42
```

Problem 5: Modular Fermat

Fermat's Last Theorem states that no three positive integers a, b, and c satisfy the equation $a^n + b^n = c^n$ for any integer value of n greater than 2.

However, for a given $n > 2$, there may exist combinations of integers a, b, and c that satisfy:

$$1 \leq a < b < c \quad \text{and} \quad a + b + c = n \quad \text{and} \quad (a^n + b^n) \bmod n = (c^n) \bmod n$$

Write a program that reads from the input a positive integer n (where $1 \leq n \leq 5000$) , and outputs the number of triples a, b, and c that satisfy the above property.

Example 1:

input:

```
100
```

output:

```
196
```

Example 2:

input:

```
1000
```

output:

```
22216
```

Example 3:

input:

```
1500
```

output:

```
27732
```