Mid-exam Imperative Programming
Oct. 5 2020, 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 two seconds.
- 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 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 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: Palindromic Number

Write a program that reads from the input a small integer n, where $1 \leq n < 10000$. The output should be the palindromic number that is obtained by appending the digits of n in reverse order to the number n.

For example, the number 123 gets appended with 321 resulting in the palindromic number 123321.

Example 1:
input:
```
123
```
output:
```
123321
```

Example 2:
input:
```
42
```
output:
```
4224
```

Example 3:
input:
```
1234
```
output:
```
12344321
```

Problem 2: Disarium Numbers

A positive integer is said to be a *Disarium number* when the sum of its digits raised to the power of their respective positions is equal to the number itself. Here, the most significant digit (i.e. first non-zero digit) has position 1.

For example, the number 598 is a Disarium number, because $5^1 + 9^2 + 8^3 = 5 + 81 + 512 = 598$.

Write a program that reads from the input an `int` n (where $n > 0$), and outputs `YES` is n if a Disarium number, and `NO` otherwise.

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

Example 2:
input:
```
42
```
output:
```
NO
```

Example 3:
input:
```
598
```
output:
```
YES
```

Problem 3: Prime Gaps

Recall that a *prime* is an integer greater than 1 that has no positive divisors other than 1 and itself.

A *prime gap* is the difference between two successive prime numbers. For example, the number 4652353 is prime, and 4652507 is the smallest prime greater than 4652353. Hence, the gap between the successive primes 4652353 and 4652507 is $154 (= 4652507 - 4652353)$.

Write a program that reads from the input an integer n (where $0 < n \leq 150$), and outputs the smallest prime p and the successive prime q such that the gap $q - p \geq n$. Also the gap size itself must be printed. The output must be in the format given in the following examples.

Example 1:
input:
```
1
```
output:
```
3-2=1
```

Example 2:
input:
```
42
```
output:
```
15727-15683=44
```

Example 3:
input:
```
150
```
output:
```
4652507-4652353=154
```

Problem 4: Swapping Letters

The input of this problem is a line with the 26 capital letters of the alphabet in some order. The first letter is at position 0, the second at position 1, and the last letter is at position 25. The following lines contain a series of actions of the form `swap x y`, where `x` and `y` are integers denoting positions. Each of these actions denotes the swapping of the letters at these positions. The series of actions is terminated by the special action `stop`.

The output of your program should be `YES` if, after applying the series of swaps, the letters are sorted in alphabetical order, and `NO` otherwise.

Example 1:

input:
```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
swap 0 3
stop
```
output:
```
NO
```

Example 2:

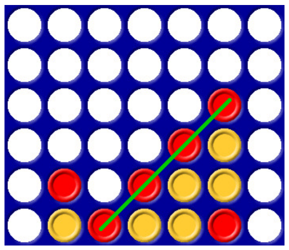
input:
```
DBCAEFGHIJKLMNOPQRSTUVWXYZ
swap 0 3
stop
```
output:
```
YES
```

Example 3:

input:
```
SDVTLMQIUFANBJRKPHOWYEXGCZ
swap 3 19
swap 0 25
swap 17 23
stop
```
output:
```
NO
```

Problem 5: Connect Four

Connect Four (also known as *Four in a Row*) is a two-player game. One player plays with yellow colored discs, and the other player plays with red colored discs. The player with the yellow discs starts the game. The players take turns dropping discs into a seven-column, six-row grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. For example, in the following figure the player with the red discs has won, because he has 4 discs in a diagonal (denoted by the green line).



We number the columns of the grid 0, 1, 2, .., 6, where column 0 is the left most column and column 6 is the right most column. We can encode the moves of a game by a series of numbers, which are the columns in which players placed a disc. The first number of this series is the starting move of the player with the yellow discs. The series is terminated by the symbol '`#`'. For example, the board configuration in the figure could be reached by the following series of moves: `3,3,1,1,4,2,4,5,5,4,5,5#`

Write a program that reads from the input a series of moves. The output of your program should be `YELLOW` if the player with the yellow discs has won the game, `RED` if the player with the red discs has won the game, or `UNDECIDED` otherwise. You may asume that the input contains a series of valid moves, so you do not have to check for invalid moves.

Example 1:

input:
```
3,3,1,1,4,2,4,5,5,4,5,5#
```
output:
```
RED
```

Example 2:

input:
```
1,2,1,2,1,2,1#
```
output:
```
YELLOW
```

Example 3:

input:
```
3,3,1,1,4,2,4,5,5,4,5,6#
```
output:
```
UNDECIDED
```