Resit Midterm Imperative Programming
Oct. 18 2021, 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: Light Numbers

A positive integer is called a *light number* if the sum of all its digits except the largest one is less than this largest digit. For example 12380 is a light number, because $1+2+3+0<8$. The number 12350 is not a light number, because $1+2+3+0>5$

Write a program that reads from the input an integer $n > 9$. The output should be `YES` if n is a light number, and `NO` otherwise.

Example 1:
input:
`12380`
output:
`YES`

Example 2:
input:
`12350`
output:
`NO`

Example 3:
input:
`42018`
output:
`YES`

Problem 2: Nested Palindromes

A positive integer is called *palindromic* if it reads the same backwards and forwards. An example of such a number is 12321.

A palindromic integer n (where $n > 10$) is called a *nested* palindrome if none of its digits is zero and its decimal representation can be split into two palindromic numbers. For example, the palindromic number 121121 can be split in the combinations $(1, 21121)$, $(12, 1121)$, $(121, 121)$, $(1211, 21)$, and $(12112, 1)$. The combination $(121, 121)$ shows that 121121 is a nested palindrome.

Note that the palindromic number 101101 is not a nested palindrome, because its decimal representation contains a zero. Another palindromic number which is not a nested palindrome is 1221, because none of the possible combinations $(1, 221)$, $(12, 21)$, $(122, 1)$ consists of two palindromic numbers.

Write a program that reads from the input an integer n, where $10 < n < 200000000$. The output should be `NESTED` if n is a nested palindrome, `PALINDROME` if n is a palindromic but not nested, and `NONE` otherwise.

Example 1:
input:
`12121`
output:
`PALINDROME`

Example 2:
input:
`121121`
output:
`NESTED`

Example 3:
input:
`4242`
output:
`NONE`

Problem 3: Lesser Emirps

A prime number p is called a *lesser emirp* if the reversal of its digits yields a larger prime. An example of a a lesser emirp is `13` since reversal of its digits yields `31`, which is a larger prime.

Write a program that reads from the input two integers a and b (where $0 \leq a \leq b \leq 5000000 = 5 \times 10^6$). The output should be the number of lesser emirps in the range a up to and including b.

Example 1:
input:
`10 20`
output:
`2`

Example 2:
input:
`2 42`
output:
`3`

Example 3:
input:
`10 1000`
output:
`18`

Problem 4: Operations

The input for this problem is an array containing integers on which you perform a set of operations in a specified order. The operations are:

1. `MULT X C`: multiply the value at index X by C,
2. `SWAP X Y`: swap the values at indexes X and Y,
3. `SUBT X C`: subtract C from the value at index X,
4. `TERM`: no further operations should be performed.

As an example, consider the array containing `[3,1,1,5,5]`. Performing the operations `MULT 2 2`, `SWAP 0 2`, `SWAP 1 2`, `SUBT 3 1` and `TERM` in that specified order leads to the array containing `[2,3,1,4,5]`.

Write a program that first reads from the input one line specifying the length of the initial array together with its values (see examples below). Next, a series of operations should be read from the input and executed on the initial array. On the output you should print the resulting array in the same format as in the following examples.

Example 1:

input:

```
5:3,1,1,5,5
MULT 2 2
SWAP 0 2
SWAP 1 2
SUBT 3 1
TERM
```

output:

```
2,3,1,4,5
```

Example 2:

input:

```
8:7,1,0,3,2,3,2,2
SWAP 0 7
SWAP 1 6
SUBT 7 7
MULT 3 3
TERM
```

output:

```
2,2,0,9,2,3,1,0
```

Example 3:

input:

```
5:4,3,2,1,0
SWAP 0 4
SWAP 1 3
MULT 2 2
MULT 3 3
MULT 4 4
TERM
```

output:

```
0,1,4,9,16
```

Problem 5: Vigenère Cipher

Encryption is a way to secure data by storing it in an unreadable format. One of the earliest encryption methods is called the *Vigenère Cipher*. In this problem we consider the encryption of a single word using a key (which is also a word). The key determines how the readable word is encrypted into an unreadable word.

The Vigenère Cipher is based on shifting letters in the alphabet. For example, if we shift the letter '`A`' one position, then we get the letter '`B`'. If we shift it two positions, we get the letter '`C`', and so on. Note that, if a letter shifts passed the end of the alphabet, then we wrap around to the letter '`A`'. So, shifting the letter '`Z`' by one position, yields the letter '`A`'.

The Vigenère Cipher works as follows. We want to encrypt the word `UNIVERSITY` using the key `PASS`. The letters in the key represent shifts of the corresponding letters in the word. The letter '`A`' denotes a shift by zero positions, the letter '`B`' denotes a shift by one position, the letter '`C`' denotes a shift by two positions and so on.

In encoding the word `UNIVERSITY`, the first letter of the word (being `'U'`) is matched with the first letter of the key. Since this is an `'P'` (which corresponds with a shift of 15), the letter `'U'` is shifted over 15 positions, yielding (after a wrap around) the letter `'J'`. The second letter of the word (being `'N'`) is matched with the second letter of the key. Since this is an `'A'` (denoting a shift by 0), the letter `'N'` stays unmodified. The letter `'I'` is shifted over 18 positions (which corresponds with the letter `'S'`) yielding the letter `'A'`. This process is repeated until all letters of the word have been encrypted.

If the key has fewer letters than the word that we want to encrypt, then the key is expanded as many times as needed by repeating itself. In the above example, the key is expanded to `PASSPASSPA` to match the length of the word `UNIVERSITY`.

Write a program that reads from the input first a line holding two integers representing the length of the word and the key. Then two lines follow holding the word and a key. On the output you should print the encrypted word.

Example 1:

input:

```
10 4
UNIVERSITY
PASS
```

output:

```
JNANTRKAIY
```

Example 2:

input:

```
11 3
PROGRAMMING
AAA
```

output:

```
PROGRAMMING
```

Example 3:

input:

```
6 3
ABCXYZ
BBB
```

output:

```
BCDYZA
```