Lab6 Learn from the past

Task

After the baptism of the five labs before, the last lab must be a piece of cake for you.

Intro

In this lab, you only need to use a high-level programming language(e.g. C/C++) to implement all the code that has been written before. **Note that the algorithm needs to be consistent with what was used before.** (e.g. Modulo operations cannot be replaced with `%` for the second lab.)

Here are program lists:

1. lab1: counting how many 1
2. lab2: a variant of the fibonacci sequence
3. lab3: longest duplicate substring
4. lab4: sort and count

rules

Here are some details:

1. You are expressly **forbidden** to use operations like ***, /, %, >>, <<** which LC3 does not support directly and the equivalent library functions;
2. You are allowed to use **+, -, =, ++, --, ==, !=, <, >, <=, >=, &, |, ~**;
3. You are allowed to use `for`, `while`, `do while`, `if`, `continue`, `break`, `switch case`;
4. You are allowed to define help functions that do not violate the above rules.

skeleton

For your convenience, your code may be written as:

```
#include <cstdint>
#include <iostream>
#include <fstream>

#define LENGTH 3
#define MAXLEN 100

int16_t lab1(int16_t a, int16_t b) {
    // initialize

    // calculation

    // return value

}

```

```
int16_t lab2(int16_t p, int16_t q, int16_t n) {
    // initialize

    // calculation

    // return value
}

int16_t lab3(int16_t n, char s[]) {
    // initialize

    // calculation

    // return value
}

void lab4(int16_t score[], int16_t *a, int16_t *b) {
    // initialize

    // calculation

    // return value
}

int main() {
    std::fstream file;
    file.open("test.txt", std::ios::in);

    // lab1
    int16_t a = 0, b = 0;
    for (int i = 0; i < LENGTH; ++i) {
        file >> a >> b;
        std::cout << lab1(a, b) << std::endl;
    }

    // lab2
    int16_t p = 0, q = 0, n = 0;
    for (int i = 0; i < LENGTH; ++i) {
        file >> p >> q >> n;
        std::cout << lab2(p, q, n) << std::endl;
    }

    // lab3
    char s[MAXLEN];
    for (int i = 0; i < LENGTH; ++i) {
        file >> n >> s;
        std::cout << lab3(n, s) << std::endl;
    }

    // lab4
    int16_t score[16];
    for (int i = 0; i < LENGTH; ++i) {

        for (int j = 0; j < 16; ++j) {
            file >> score[j];
        }
        lab4(score, &a, &b);
        for (int j = 0; j < 16; ++j) {
            std::cout << score[j] << " ";
        }
        std::cout << std::endl << a << " " << b << std::endl;
    }

    file.close();
    return 0;
}
```

with the test.txt we provide, here is the output

```
2
4
15
146
818
1219
3
4
3
0 10 20 25 30 35 40 45 50 55 60 80 85 90 95 100
4 1
0 10 15 20 25 35 40 45 50 65 70 75 80 90 95 100
3 2
9 10 11 21 22 33 44 53 55 57 66 77 88 97 98 99
4 1
```

Note:

1. If you use the programming framework we provide, for the convenience of TA's test, please comment out the `#define LENGTH 3` when submitting. (So TA can use `-DLENGTH=X` since there are more testcases.)
2. If you write from scratch yourself, please describe your program structure in the report, and make sure your output is consistent with our skeleton.

Score

Correctness for 50% and the report for other 50%.

Submission

Your submission be structured as shown below.

```
PB21******_Name.zip
├── PB21******_Name_report.pdf
└── lab6.c
```

Reports

Your reports should contain at least the five parts below:

- purpose
- principles (e.g. how to solve these problems using high-level programming language)
- procedure (e.g. bugs you encountered and how to solve them)
- result of your test
- answers to the questions
    - What is the difference between programming in a high-level language and programming in LC3 assembly language?
    - What instructions do you think need to be added to LC3? (Hint: You can think about the previous labs and what instructions could be added to greatly simplify the previous programming)
    - Is there anything you need to learn from LC3 for the high-level language you use?