Lab 6 Learn from the past

Task

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

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 and you cannot use `strcmp()` in lab3)
Here are program lists:

1. lab1: counting zero
2. lab2: the pingpong sequence
3. lab3: password
4. lab4: baguenaudier

Rules

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 can only use certain data types, including `int`, `int16_t`, `char` and pointers/arrays of the same type.
5. 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>
#include <bitset>


#define LENGTH 1
#define MAXLEN 100
#define STUDENT_ID_LAST_DIGIT 3



int16_t lab1(int16_t n) {
    // initialize

    // calculation

    // return value
}

int16_t lab2(int16_t n) {
    // initialize

    // calculation

    // return value
}

int16_t lab3(char s1[], char s2[], int input_cnt, char my_input
    // initialize

    // calculation

    // return value
}
```

```
int16_t lab4(int16_t *memory, int16_t n) {
    // initialize

    // calculation

    // return value
}


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



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

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

    // lab3
    std::cout << "===== lab3 =====" << std::endl;
    char passwd[MAXLEN]; char verify[MAXLEN];
```

```
    int input_cnt=-1;
    char my_input[10][MAXLEN];
    for (int i = 0; i < LENGTH; ++i) {
        file >> passwd >> verify;
        file >> input_cnt;
        for (int j=0; j< input_cnt; j++)
        {
            file >> my_input[j];
        }
        std::cout << lab3(passwd, verify , input_cnt, my_input)
    }

    // lab4
    std::cout << "===== lab4 =====" << std::endl;
    int16_t memory[MAXLEN], move;
    for (int i = 0; i < LENGTH; ++i) {
        file >> n;
        int16_t state = 0;
        move = lab4(memory, n);
        for(int j = 0; j < move; ++j){
            std::cout << std::bitset<16>(memory[j]) << std::endl
        }
    }



    return 0;
}
```

you can run using g++

```
rm lab6
rm lab6.o
g++ -g -c lab6.cpp -o lab6.o
g++ -g lab6.o -o lab6
```

```
chmod 777 lab6
./lab6
```

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

```
===== lab1 =====
17
===== lab2 =====
786
===== lab3 =====
wron
world
wron
===== lab4 =====
0000000000000001
0000000000000101
0000000000000100
0000000000000110
0000000000000111
```

Note:

1. If you use the programming framework we provide, for the convenience of TA's test, please comment out the `#define LENGTH 1` when submitting. (So TA can use `DLENGTH=X` since there are more testcases, we provide `test_multi.txt` to help you understand this process.)
2. Since we used the student number for calculations and checksums in lab1, in this lab you will also need to set the last digit of your student number by modifying the macro definition `STUDENT_ID_LAST_DIGIT`.
3. If you write from scratch yourself, please describe your program structure in the report, and make sure your output is consistent with our skeleton.
4. the input of lab3 in test.txt:

```
hello world 2 abcde fghij
```

Here hello is the password, world is the verification, the following figure is the number of inputs.

Additional Requirements

If you don’t comply with these requirements, the lab may be counted as an invalid work.

1. Your report should be structured into the following sections:
    - Purpose
    - Principles
    - Procedure (e.g. bugs or challenges you encountered and how to solve them)
    - Results
2. Your submission be structured as shown below.

```
PB********_Name.zip
├── PB********_Name_report.pdf
└── lab6.c/cpp
```