UEE1303(1070) S12: Object-Oriented Programming

Lab 02: Exercise

What you will learn from Lab 2

In this laboratory, you will practice problem solving and review the programming still.

*TASK 2-1 EVALUATION SYSTEM

- ✓ Please write an evaluation system to analysis the student midterm online test for UEE1302 course. The evaluation system first reads the student scores from an input file, called "`score.txt`" and outputs the analysis results to file "`result.txt`". The command-line usage of the evaluation system is

```
>./ex2-1 score.txt result.txt
```

- ✓ In the input file "`score.txt`", each row indicates a record for one student. As shown in the sample file "`ex2-1.in`", the first and second columns denote the student's ID and name, respectively. Since there are 9 problems in the online test, the students can select the problem set instead of writing every problem. However, the scores for different problems are different. The problems can be classified into the easy level, medium level and hard level. The first three problems are easy-level. If the student correctly answers one of easy-level problems, he/she can obtain 10 points. The next three problems are medium-level and 20 points are given for each one. The last three problems are hard-level and one can get 30 points for one correct answer. The full score is 100. The last column indicates the required time that the student returns the answer sheet.

```
01 Tom 1 1 1 0 0 1 0 0 1 180
02 Jean 1 1 0 1 0 1 0 1 0 150
03 Kevin 0 0 0 1 0 1 0 1 1 90
04 John 1 0 0 0 0 0 1 1 1 70
05 Marry 1 1 1 1 1 1 1 0 0 150
```

- ✓ The evaluation system needs to calculate the final score for every student and sort the results according to the final score and the required time. The list is sorted in a decreasing order depending on the final scores first. If two students have the same final scores, the student who requires less time has a higher rank. The sample output file "`ex2-1.out`" is shown as follows.

```
04 John 100 70
03 Kevin 100 90
05 Marry 100 150
02 Jean 90 150
01 Tom 80 180
```

✓ Note that you have to design the data structure called `student` to store the information for every student.

TASK 2-2 BIG NUMBER OPERATION

✓ Big Number calculation is frequently seen while dealing with a large number. For example, there is no built-in data type can store a number with 100 digits. To solve the problem, you have to write a program to process the large number digit-by-digit and store digits in an integer array.

✓ You have to process input file "`bignumber.txt`" to obtain two big numbers and output the results of arithmetic operations to "`result.txt.`"

The command-line usage of the program is shown below

```
>./ex2-2 bignumber.txt result.txt
```

The required format for a sample input file "`ex2-2.in`" as "`bignumber.txt`" is shown below. In the "`bignumber.txt`" file, the first line and the second line indicate the big number `A` and `B`, respectively.

```
12443
5467755
```

Your result should be written to a user-specified output file. The sample output for "`ex2-2.in`" is file "`ex2-2.out`" with the content shown as below.

```
5480198   // A + B
-5455312  // A - B
```

✓ Requirement

➢ Main function:

```
// ex2-2.cpp
```

```
typedef struct
{
    int *data;
    int length;
    bool sign;
}BIGNUMBER;

int main(int argc, char *argv[])
{
    BIGNUMBER a, b;
    ReadTextFile(argv[1], a, b);

    BIGNUMBER results[2];

    results[0] = BigNumberOperation(a,b,'+');
    results[1] = BigNumberOperation(a,b,'-');

    WriteResults(argv[2], results);


    return 0;
}
```

You have to finish this exercise using the above structure and three more functions of your own: `ReadTextFile()`, `BigNumberOperation()` and `WriteResults()`.

- ✓ Hint
  - ➢ Pre-processing:
    1. Initialize the character array
    2. Change the integer to character array in revised order.

       `A = 12443; B = 5467755`

a.data[0]	a.data[1]	a.data[2]	a.data[3]	a.data[4]	a.data[5]	a.data[6]
3	4	4	2	1	0	0

b.data[0]	b.data[1]	b.data[2]	b.data[3]	b.data[4]	b.data[5]	b.data[6]
5	5	7	7	6	4	5

- Addition:

```
for (i = 0; i < max_length ;i++)
    c.data[i] = a.data[i] + b.data[j];
```

c.data[0]	c.data[1]	c.data[2]	c.data[3]	c.data[4]	c.data[5]	c.data[6]
8	9	11	9	7	4	5

If the digit is larger than 10, then the overflow is added to next digit.

c.data[0]	c.data[1]	c.data[2]	c.data[3]	c.data[4]	c.data[5]	c.data[6]
8	9	1	0	8	4	5

- Print-out: `PrintNumber()`

  Print the array in a reversed order starting from non-zero digit.