Project 1: Sudoku Puzzle

1 Description

The objective of this project is to gain a deeper insight into `functions, array(or vector)`, and `file operation`. Especially `file operation`, we provide skeleton codes, so be familiar with it. In this assignment, you need to implement multiple functions for solving Sudoku. Your program should read data from the file(`input.txt`) and produce the output according to the instructions below:
We will evaluate your program with N? test input files, each of which is partially filled with a `9x9` Sudoku board. If your program produces correct answers for all the inputs, you will receive 90 points (i.e., N pts/input). If your output is wrong for each input file, you will receive 0. The remaining 10 points are report scores. You should write a report on how your code works as detailed as you can.

Task 1: Determine Valid Sudoku

Task 1 is the problem of determining whether Sudoku is valid. This function gets a `9x9` size Sudoku as input and It is partially filled. We will determine whether Sudoku received as input is valid or not according to the rules. Rules to consider are as follows:

1. Each row must contain the digits `1-9` without repetition.
2. Each column must contain the digits `1-9` without repetition.
3. Each of the nine `3 x 3` sub-boxes of the grid must contain the digits `1-9` without repetition.

Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.
- The size of the input Sudoku board is always `9x9`. The board that does not satisfy this range is not given.
- In a given input board, the number has a value between `0` and `9`. A zero means a blank.

Task 2: Implement Sudoku Solver

Task 2 is the problem of writing a program to solve a Sudoku puzzle by filling the empty cells. A sudoku solution must satisfy all of the following rules:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. Each of the digits `1-9` must occur exactly once in each of the 9 `3x3` sub-boxes of the grid.

Note:

- We will only consider a solvable Sudoku board.

2 Instruction: Input File

The input file is a 9-line `.txt` file.
In each row, 9 numbers ranging from 0 to 9 are listed without spaces.

Sample Input 1

In "Input1.txt",

```
000100435
700002001
004053000
000241370
130905046
092736000
000610800
200500003
681004000
```

Sample Input 2

In "Input2.txt",

```
000100445
700002001
004053000
000241370
130905046
092736000
000610800
200500003
681004000
```

3 Instruction: Output File

Output files are automatically generated according to skeleton code. Therefore, there is no need to make an effort to create an output file. However, it is necessary to look at the output file to determine whether your code has been implemented accurately.
If the Sudoku of the input is valid, the Sudoku filled in the empty part(`0`) will be filled and output according to Task2. On the other hand, if the Sudoku of the input is invalid because it does not satisfy the rules, an error message will be output.

Sample Output 1: "Output1.txt"

In "output1.txt", valid case

```
826197435
753462981
914853762
568241379
137985246
492736158
375619824
249578613
681324597
```

Sample Output 2: "Output2.txt"

In "output2.txt", invalid case.

```
This is invalid sudoku
```

4 Compilation instruction

How to compile and run the c++ program on your local machine

0. open a new terminal window or cmd if you are on windows.
1. change the directory to the directory in which you have `main.cpp` file. For example, if it is in `/Users/kimsolang/Desktop/eep` (Mac) or `C:/Users/kimsolang/eep` (Windows), enter your command line:

   ```
   $ cd /Users/kimsolang/Desktop/eep
   ```

   or

   ```
   $ cd 'C:/Users/kimsolang/eep'
   ```

2. Now enter the following command to compile the source file using g++.

   ```
   $ g++ -o <name-you-want-to-give> main.cpp --std=c++14
   ```

   In place of <name-you-want-to-give> replace it with any name like `test`, etc.
3. (**General case**) Run it! Now you can run the program using:

   ```
   $ ./<name-you-want-to-give>
   ```

   or

   ```
   $ ./test
   ```

4. (**Important**) In this project, we use an input file and output file to run

   ```
   $ ./<name-you-want-to-give> <input-file-name> <output-file-name>
   ```

   or

   ```
   $ ./test input1.txt output1.txt
   ```

How to compile and run the C++ program in OnlineGDB

1. Set Language to "C++ 14"
2. Click "Upload File" and upload your input file
3. Implement your program in the `main.cpp` (before starting implementation, copy and paste skeleton code in `main.cpp`)
4. Write `input.txt output.txt` (<input-file-name> <output-file-name>) at the bottom of the window (**Command line arguments**).
5. Click "Run"

5 File Format

Your main file name should be `main.cpp` and you should write your student id and your name in your main file.

```
// StudentId YourName
#include <iostream>
Your Code Here . . .
```

6 Submission and Report

Submission

You should submit your code and a report on the blackboard. Please format your submission as a single zip file 'StudentID YourName.zip' that includes the files of your source code and the report,
e.g.,

```
20225067_SolangKim.zip
- main.cpp
- report.pdf
```

Late Submission

We allow late submission, but we will give a penalty of 20% every day. This is only for 2 days and will be given zero points from then on.

- If you submit it within 24 hours after the date, you will be deducted 20 percent from the total score received.
- If you submit it within 24 to 48 hours, you will be deducted 40 percent from the total score received.
- If you submit it after 48 hours, 0 points.

Report

For the report, write the most detailed explanation you can do. The report should include:

- How the function you implement works (make comment on each line)
- Describe the sample input and output of your program
- Any format except handwriting and submit it in **pdf**.
- The maximum length of the report is **3 pages.**

Important

- **Cheating is never allowed. If cheating is found, you will get zero points in this project. There is NO MERCY.**
- Libraries that are not covered in class **cannot be used**.
- Don't edit the skeleton code of the part that makes the output because of evaluating. Other parts can be changed for your convenience.
- You can make and use functions other than the given functions in skeleton code. (Probably necessary)