The mission of this assignment is to implement a "Sudoku game board" (https://en.wikipedia.org/wiki/Sudoku) and interact with user using a small set of commands that enable to initialize the game board. You program must permit user to see the game board, swap rows, columns, or areas of the board for randomization, verify the correctness of the board content, and erase values of the generated game board.  
   
 User can enter the following : show, swap, verify, erase, and quit command:  
   
 'show' :will display the board content  
 'swap' : will exchange some randomly generated row/column position with another row/column position - note that a simple swap only occurs within a permissible range (see example below)  
 'verify' : will check all the consistencies enforced for the Sudoku game and print out which places are wrong  
 'erase' : will pick up a random cell of the Sudoku board and place that cell as '-' mark visible to user  
 'quit' :will stop execution  
 other input must show an error message and ask to reenter a correct command  
 Implementation requires you to utilize a structure holding "vector<vector<short> >" to represent the 9 x 9 game board. The main program is a mere loop to dispatch the execution of each read command. This submission will be somewhat equivalent to the midterm exam of this course work. Note that there will be no extension of due in order to proceed for the FLTK and Graphic/GUI subject.   
   
 The next example tells what are required to realize:  
   
 Welcome Sudoku Initializer!  
 > show  
 A B C D E F G H I   
 P 2 3 4 8 9 1 5 6 7  
 Q 5 6 7 2 3 4 8 9 1  
 R 8 9 1 5 6 7 2 3 4   
 S 3 4 5 9 1 2 6 7 8   
 T 6 7 8 3 4 5 9 1 2   
 U 9 1 2 6 7 8 3 4 5   
 V 1 2 3 7 8 9 4 5 6   
 W 4 5 6 1 2 3 7 8 9   
 X 7 8 9 4 5 6 1 2 3   
 > swap  
 Trying to swap rows Q and S...   
 - Rows P and S were swapped...   
 - Rows Q and T were swapped...   
 - Rows R and U were swapped...  
 > show   
 A B C D E F G H I   
 P 3 4 5 9 1 2 6 7 8   
 Q 6 7 8 3 4 5 9 1 2   
 R 9 1 2 6 7 8 3 4 5   
 S 2 3 4 8 9 1 5 6 7   
 T 5 6 7 2 3 4 8 9 1   
 U 8 9 1 5 6 7 2 3 4   
 V 1 2 3 7 8 9 4 5 6   
 W 4 5 6 1 2 3 7 8 9   
 X 7 8 9 4 5 6 1 2 3   
 > swap  
 Trying to swap columns H and G...   
 - Columns H and G were swapped...  
 > show  
 A B C D E F G H I   
 P 3 4 5 9 1 2 7 6 8   
 Q 6 7 8 3 4 5 1 9 2   
 R 9 1 2 6 7 8 4 3 5   
 S 2 3 4 8 9 1 6 5 7   
 T 5 6 7 2 3 4 9 8 1   
 U 8 9 1 5 6 7 3 2 4   
 V 1 2 3 7 8 9 5 4 6   
 W 4 5 6 1 2 3 8 7 9   
 X 7 8 9 4 5 6 2 1 3   
 > verify  
 - All columns, rows, and components are OK...  
 > erase  
 Erasing row P column F  
 > erase  
 Erasing row X column A  
 > show  
 A B C D E F G H I   
 P 3 4 5 9 1 - 7 6 8   
 Q 6 7 8 3 4 5 1 9 2   
 R 9 1 2 6 7 8 4 3 5   
 S 2 3 4 8 9 1 6 5 7   
 T 5 6 7 2 3 4 9 8 1   
 U 8 9 1 5 6 7 3 2 4   
 V 1 2 3 7 8 9 5 4 6   
 W 4 5 6 1 2 3 8 7 9   
 X - 8 9 4 5 6 2 1 3   
 > verify   
 - Found inconsistency in row P...   
 - Found inconsistency in row X...   
 - Found inconsistency in column A...   
 - Found inconsistency in column F...   
 - Found inconsistency in component starting at row P and column D...   
 - Found inconsistency in component starting at row V and column A...  
 > quit  
 Bye...  
   
 This project does not require to implement a value input. However for fun, that part can be added. The input logic for parsing a triple value (row, column and value) becomes more complicated and so a professional version for this kind of parsing is attached for reference below :

**Parsing reference:**

\* Input parsing logic for this application. This one uses so-called "regular   
 expressions" to prune out inconsistent input (i.e., syntactic check).   
 One word command such ans show, quit, etc., and three word input (row,   
 column, and value triple) such as Q A 3, qa3, q a3, etc are parsed. Use   
 C++11 standard to compile this example. \*/  
#include <iostream>  
#include <cstdlib>  
#include <cassert>  
#include <sstream>  
#include <algorithm>  
#include <regex>  
using namespace std;  
  
// enum gives a way to define explicitly named constants (e.g., Sunday, Monday,  
// January, February, etc. The next is to define command set of this example.  
enum class Command { SHOW, SWAP, VERIFY, ERASE, FILL, QUIT };  
  
Command parse(char& row, char& col, int& num) {  
 cout << "\nEnter show/swap/verify/quit command or row column and value: ";  
 string line;  
 regex pattern1("show|swap|verify|erase|quit"); // one word input.  
 regex pattern2("[p-x][[:space:]]\*[a-i][[:space:]]\*[1-9]"); // 3 word input.  
 while (getline(cin, line)) { // read a whole line and convert to lower cases.  
 transform(line.begin(), line.end(), line.begin(), ::tolower);  
 if (regex\_match(line, pattern1)) { // match one word input?   
 if (line.find("show") != string::npos) return Command::SHOW;  
 if (line.find("swap") != string::npos) return Command::SWAP;  
 if (line.find("verify") != string::npos) return Command::VERIFY;   
 if (line.find("erase") != string::npos) return Command::ERASE;   
 if (line.find("quit") != string::npos) return Command::QUIT;   
 }   
 if (regex\_match(line, pattern2)) { // match 3 word input?   
 istringstream iss(line); // then read into 3 parts.  
 iss >> row >> col >> num; // stringstream is the one to use!  
 assert(!iss.fail()); // this will not happen (why?)  
 return Command::FILL;   
 }  
 cout << "Input error, try again: ";   
 }  
 assert(false); // should not reach here.  
}  
  
int main() {  
 char row, col; int val;  
  
 while (true) {  
 switch (parse(row, col, val)) {  
 case Command::SHOW: cout << "show understood..." << endl; break;  
 case Command::SWAP: cout << "swap understood..." << endl; break;  
 case Command::VERIFY: cout << "verify understood..." << endl; break;   
 case Command::ERASE: cout << "erase understood..." << endl; break;  
 case Command::QUIT: cout << "quitting..." << endl; goto END;  
 case Command::FILL: cout << "(" << row << ", " << col << ") <- " << val << endl; break;  
 default: assert(false); // should not reach here.  
 }  
  
 }  
 END:   
 cout << "Bye..." << endl;  
 return 0;  
}