1: Sudoku Helper Vision Statement and State Class

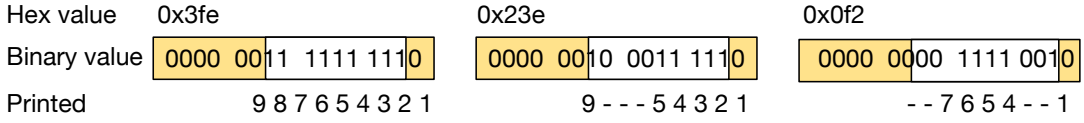
CSCI 6626 / 4526 Fall 2019

1 Instructions: One Square on a Sudoku Board

You have to start somewhere, and the logical starting place is the lowest-level data structure, the square. Each square has a state and a set of relationships to other squares. This first assignment asks you to implement the State of a square. More parts (data and functions) will be added to this class as the project develops.

Define a class State. Implement the following function members:

- It should have these data members, in the order given: possibilities (short), value (char), and fixed (boolean). A value is fixed if it was part of the original puzzle read from the input file. Ask in class about why the order is important.
- A default null constructor.
- A constructor with a char parameter. Initialize the value to the parameter (a dash or a single digit). If the value is a digit, initialize the possibility list to 0 and the fixed flag to true. If it is a '-', initialize the fixed flag to false and the possibility list to 0x3fe (In binary, 0000 0011 1111 1110.) This means that all nine digits are still possible for this square. (Later assignments will change this list of possibilities.)
- A default destructor.
- mark(char ch) : Print an error comment if this State is fixed, and do not mark it. Otherwise, make ch the new value in the State. Do not abort.
- A method for `print()` that prints all data in the State in a readable format. Print the possibility list as a series of digits and dashes. For example, if the value was 0x11e, you would print `12345---9`. Do this in a loop by right-shifting the possibility list one bit and masking off the 1's bit, then testing it. Example output is shown below for three different possibility lists.



- Outside the class but inside the .hpp file, declare an inline method for the output operator. It must call your print() function with the appropriate parameter. This definition allows you to output a State as easily as you output an integer or a string.

2 Testing and Submission

Due September 3. Write a main program and a first version of a unit test for this class. Call the unit test from main. In the unit test, call each of the functions defined in the State class and verify that they work properly. Submit evidence that they work. This will test the default possibility lists. To test a different possibility list, change your program (temporarily) to initialize the possibilities to 0x23e or 0x0f2.

3 Advice

This is a very short, very easy program. Don't complicate it. Its purposes are to make sure you understand

- The technical vocabulary
- How to organize and use a multi-module project,
- How to define a class with header and implementation files
- How to do the easiest kind of bit operations.

Email for help promptly if you are confused about the instructions or you have any trouble of any sort. Be sure you have something to hand in by the due date. If you finish this early, hand it in early and start on Program 3: Implementing Square. Instructions will follow.