```
1  //------------------------------------------------------------------------
2  #ifndef SUDOKUH
3  #define SUDOKUH
4  //------------------------------------------------------------------------
5  #include <stddef.h> // size_t
6
7  class Sudoku
8  {
9    public:
10       // Used by the callback function
11     enum MessageType
12     {
13       MSG_STARTING,      // the board is setup, ready to go
14       MSG_FINISHED_OK,   // finished and found a solution
15       MSG_FINISHED_FAIL, // finished but no solution found
16       MSG_ABORT_CHECK,   // checking to see if algorithm should continue
17       MSG_PLACING,       // placing a symbol on the board
18       MSG_REMOVING       // removing a symbol (back-tracking)
19     };
20
21       // 1-9 for 9x9, A-P for 16x16, A-Y for 25x25
22     enum SymbolType {SYM_NUMBER, SYM_LETTER};
23
24     const static char EMPTY_CHAR = ' ';
25
26       // Implemented in the client and called during the search for a solution
27     typedef bool (*CALLBACK)
28       (const Sudoku& sudoku, // the gameboard object itself
29        const char *board,    // one-dimensional array of symbols
30        MessageType message,  // type of message
```

```
        size_t move,          // the move number
        unsigned basesize,    // 3, 4, 5, etc. (for 9x9, 16x16, 25x25, etc.)
        unsigned index,       // index of current cell
        char value            // symbol (value) in current cell
       );

    struct SudokuStats
    {
      unsigned basesize; // 3, 4, 5, etc.
      unsigned placed;   // the number of values the algorithm has placed
      size_t moves;      // total number of values that have been tried
      size_t backtracks; // total number of times the algorithm backtracked
      SudokuStats() : basesize(0), placed(0), moves(0), backtracks(0) {}
    };

      // Constructor
    Sudoku(int basesize, SymbolType stype = SYM_NUMBER, CALLBACK callback = 0);

      // Destructor
    ~Sudoku();

      // The client (driver) passed the board in the values parameter
    void SetupBoard(const char *values, size_t size);

      // Once the board is setup, this will start the search for the solution
    bool Solve();

      // For debugging with the driver
    const char *GetBoard() const;
    SudokuStats GetStats() const;

  private:
    size_t moves_;

    // Other private fields and methods...
    char* board_;
    size_t board_len_;
    //size_t move_;
    SymbolType stype_;
    SudokuStats stats_;
    CALLBACK callback_;

    bool place_value(unsigned x, unsigned y);
    bool is_valid(unsigned x, unsigned y, char val);
};

#endif  // SUDOKUH
```