```cpp
//============================================================================
// Name        : Sudoku.cpp
// Version     : 1.0
// Copyright   :
// Description : Solver of SUDOKU game
//============================================================================
/*******************************************************************************************************************
Programmed by: Luis Barquero
Purpose: Program will read in a text file containing an unsolved Sudoku problem and it will recursively solve it.
*******************************************************************************************************************/

#include <iostream>
#include <string>
#include "RecursiveSolver.h"
#include "StackSolver.h"
#include <omp.h>
//#include "Solver.h"

using namespace std;

int main() {
    string filename;
    cout << "Please type in the problem file name: ";
    cin >> filename;
    cout << endl;

    cout << "Please select a solver:" << endl;
    cout << "\t1. Recursive Solver" << endl;
    cout << "\t2. Stack Solver" << endl;
    cout << "What is your choice? (1/2):";
    int choice;
    cin >> choice;
    int tn = 2; //the number of threads
    omp_set_num_threads(tn);
    if (choice == 1) {
        // Create a recursive solver
        RecursiveSolver rSolver;
        int empty_slot_number = rSolver.read_problem(filename);

        // Solve the game. If solved, print the answer
        if (empty_slot_number <= 0)
            cout << "No empty slot to be filled in!" << endl;
        else
        {
            #pragma omp critical
            for(int i = 0; i < tn; i++)
            {
                clock_t start = clock();
                rSolver.solve(0);
                cout << "\nThread " << i + 1 << " Elapsed time: " << ((double)clock() - start) / CLOCKS_PER_SEC << endl;
                cout << "\n" << endl;
            }
        }

        // No answer
        if (!rSolver.solved)
            cout << "There is no answer for this problem!" << endl;
    }
    else if (choice == 2) {
        // Create a stack solver
        StackSolver sSolver;
        int empty_slot_number = sSolver.read_problem(filename);

        // Solve the game. If solved, print the answer
        if (empty_slot_number <= 0)
            cout << "No empty slot to be filled in!" << endl;
        else
            sSolver.solve();

        // No answer
        if (!sSolver.solved)
            cout << "There is no answer for this problem!" << endl;
    }
    else
        cout << "Invalid choice! Please input 1 or 2!" << endl;

    return 0;
}

```