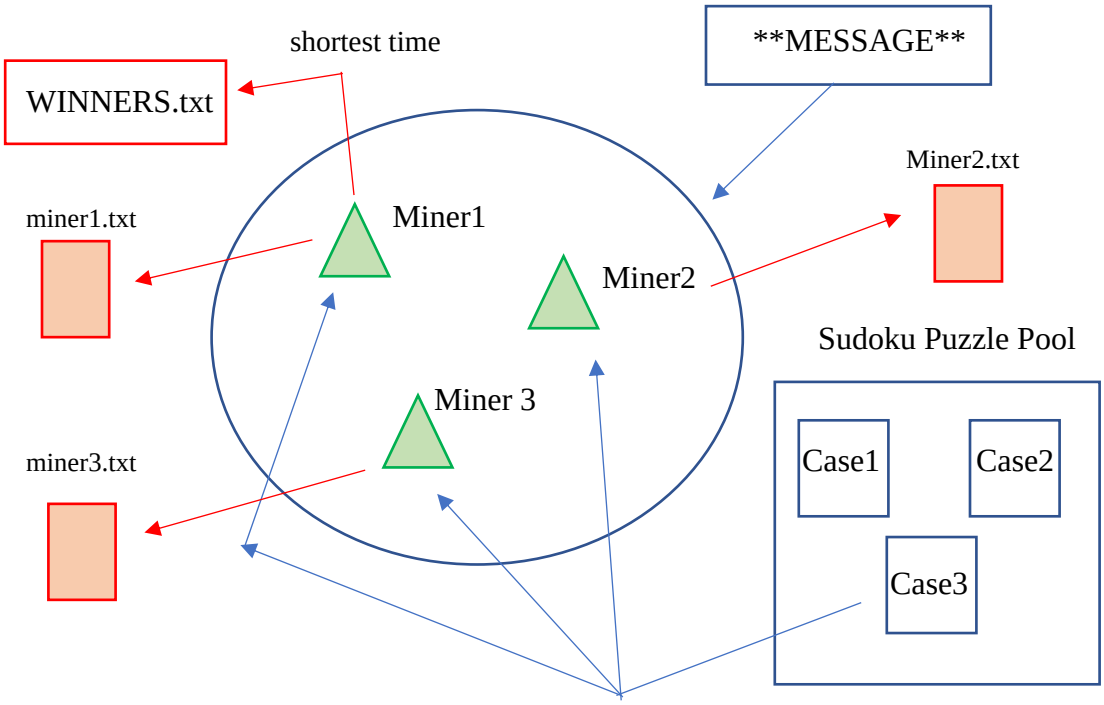
Data Structures (CS-2001) Documentation

Project Task-1 (BCS-C)

- Rayyan Minhaj 20K-0143
- Kainat Afzal 20K-0281
- Hafiza Sara 20K-1749
- Tahreem Fatima 20K-0483



Randomly Selected Test Case

We have created 3 miners (i.e., 3 separate Miner functions) that take in our message as parameter

```
80   void miner1(string x) { ... }
123
124
125  void miner2(string x) { ... }
163
164
165  void miner3(string x) { ... }
202
```

After that we create 3 different threads (m1, m2 and m3) and pass each miner function pointer to the constructor of thread class object along with our string message. Then we joined each of those 3 threads using thread::join( ) thus applying multi-threading concept.

```
int main()
{
    string x = "***MESSAGE***";

    thread m1(miner1,x);
    thread m2(miner2,x);
    thread m3(miner3,x);

    m1.join();
    m2.join();
    m3.join();
```

Similarly, inside our Miner function we have implemented a method of finding the exact time taken (in millisecond) for function execution i.e., how much time it took to solve the sudoku puzzle

```
void miner1(string x) {
    vector<int> values(10000);
    auto f = []() -> int { return rand() % 10000; };
    generate(values.begin(), values.end(), f);
    auto start = high_resolution_clock::now();
    sort(values.begin(), values.end());



    sudoku_solver();


    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);

    t1 = duration;


}
```

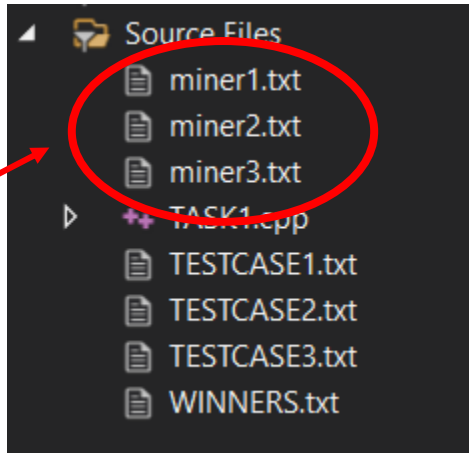
Inside each miner function we are calling the Sudoku_Solver( ) func

Whichever miner completes the sudoku puzzle test case first, logs its ID, message and the time taken to solve it inside a txt log file (WINNERS.txt) inside main( ) by determining which miner took the least time to solve sudoku (global variables)

The WINNERS.txt file is as shown below. Which miner's data to write in the file is determined by the **time** (milliseconds) which is stored in 3 global variables.

```
MINER 3 ***MESSAGE*** 133221 milliseconds
MINER 3 ***MESSAGE*** 24458359 milliseconds
MINER 1 ***MESSAGE*** 35657734 milliseconds
MINER 2 ***MESSAGE*** 28528361 milliseconds
```

Likewise, inside main we log each individual file of the miner as well with their respective names (MINER 1, MINER 2 or MINER 3), the message and the time taken



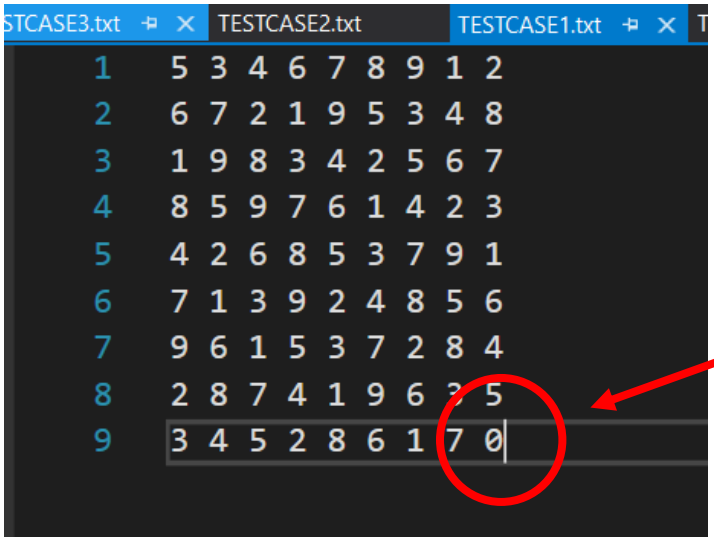

```
MINER 1 ***MESSAGE*** 133321 milliseconds
MINER 1 ***MESSAGE*** 24458540 milliseconds
MINER 1 ***MESSAGE*** 35657734 milliseconds
MINER 1 ***MESSAGE*** 28528789 milliseconds
```

Using those global variables inside main, we are firstly, appending the file of each miner and then secondly, determining which miner took the minimum time

```
fstream file4;
file4.open("WINNERS.txt", std::ios_base::app | std::ios_base::in);


if (t1.count() <= t2.count() && t1.count() <= t3.count()) {
    file4 << "MINER 1 " << x << " " << t1.count() << " milliseconds\n";

}
else if (t2.count() <= t1.count() && t2.count() <= t3.count()) {
    file4 << "MINER 2 " << x << " " << t2.count() << " milliseconds\n";

}
else if (t3.count() <= t2.count() && t3.count() <= t1.count()) {

    file4 << "MINER 3 " << x << " " << t3.count() << " milliseconds\n";

}

file4.close();
```

Coming to the Sudoku_Solver( ) function, we have implemented a brute-force approach of finding the very last value of a 9x9 Sudoku puzzle.

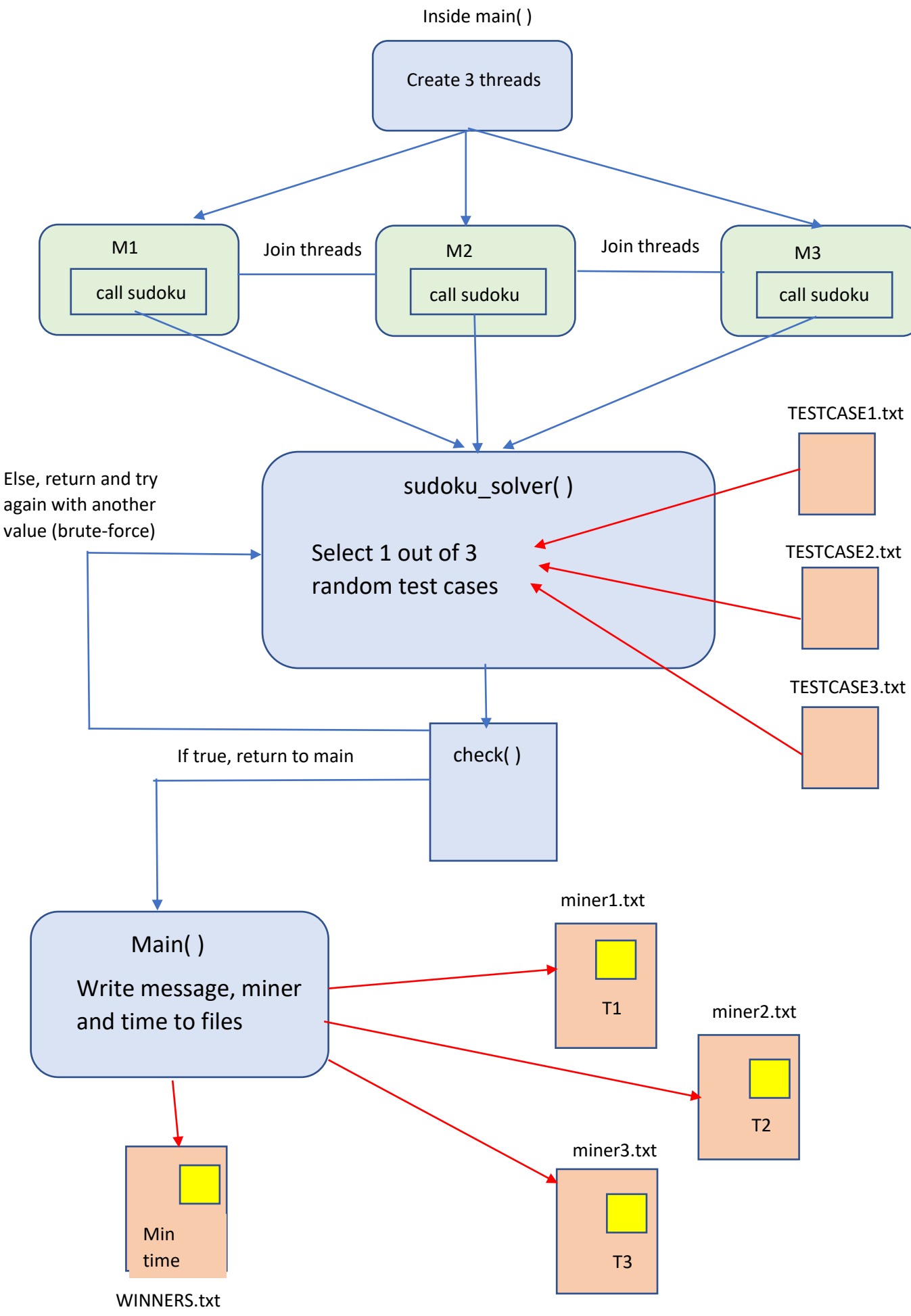
We randomly keep placing a single digit between 1-9 at the last (highlighted below) index of the Sudoku matrix until the value comes correct (which is then verified by the community, a bool check( ) function)



Brute-forcing a random integer over here until correct output

The checker function we have implemented

```
bool check(int** a) {
    int sum=0;
    for (int j = 0; j < 9; j++) {
        sum += a[8][j];
        //if(j==8){
        //    a[i][8]=45-sum;
    }
    //    a[8][j]=45-sum;


    if (sum == 45) {
        return true;
    }
}
```


Inside main( )
Create 3 threads
M1
call sudoku
Join threads
M2
call sudoku
Join threads
M3
call sudoku
TESTCASE1.txt
TESTCASE2.txt
TESTCASE3.txt
Else, return and try again with another value (brute-force)
sudoku_solver( )
Select 1 out of 3 random test cases
check( )
If true, return to main
Main( )
Write message, miner and time to files
miner1.txt
T1
miner2.txt
T2
miner3.txt
T3
Min time
WINNERS.txt