Multithreading Assignment — Sudoku Solution Validator

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

Figure 1.0 Solution to a 9×9 Sudoku puzzle.

A **Sudoku puzzle** uses a 9×9 grid in which each column and row, as well as each of the nine 3×3 subgrids, must contain all of the digits 1···9. *Figure 1.0* presents an example of a valid Sudoku puzzle. This exercise consists of designing a multithreaded application that determines whether the solution to a Sudoku puzzle is valid.

There are several different ways of multithreading this application. One suggested strategy is to create **threads** that check the following criteria:
- A thread to check that <u>each column</u> contains the digits 1 through 9
- A thread to check that <u>each row</u> contains the digits 1 through 9
- Nine threads to check that <u>each of the 3×3 subgrids</u> contains the digits 1 through 9

This would result in a total of **eleven separate threads** for validating a Sudoku puzzle. However, you are welcome to create even more threads for this exercise. For example, rather than creating one thread that checks all nine columns, you could create nine separate threads and have each of them check one column.

Passing Parameters to Each Thread

The parent thread will create the worker threads, passing each worker the location that it must check in the Sudoku grid. This step will require passing several parameters to each thread. The easiest approach is to create a data structure using a **struct**. For example, a structure to pass the row and column where a thread must begin validating would appear as follows:

```
/* structure for passing data to threads */
typedef struct
{
    int row;
    int column;
} parameters;
```

Pthreads will create worker threads using a strategy similar to that shown below:

```
parameters *data = (parameters *) malloc(sizeof(parameters));
data->row = 1;
data->column = 1;
/* Now create the thread passing it data as a parameter */
```

The **data** pointer will be passed to the **pthread_create()***(Pthreads)* function which in turn will pass
it as a parameter to the function that is to run as a separate thread.

Returning Results to the Parent Thread

Each worker thread is assigned the task of determining the validity of a particular region of the Sudoku puzzle. Once a worker has performed this check, it must pass its results back to the parent. One good way to handle this is to create an **array of integer values** that is visible to each thread. The ith index in this array corresponds to the ith worker thread. If a worker sets its corresponding value to 1, it is indicating that its region of the Sudoku puzzle is valid. A value of 0 would indicate otherwise. When all worker threads have completed, the parent thread checks each entry in the result array to determine if the Sudoku puzzle is valid.