Rationale:

Data collision is when two or more different sources are trying to access to same data at the same time. While the data is processed by one source, the other source accesses the data and changes its value. This will cause the other source to work the wrong value which eventually gives a wrong data. In this part of the assignment, itwill test whether data collisions can occur when multiply threads are accessing the same data. Since threads are shared the same data with other threads in the same process, it mostly like to the cause a data collision. The goal of this assignment is going to be writes a program with multiple threads accessing the same data and compare the calculated final value with the expected value.

Method:

The program has three different classes, one class is the main class and the rest are thread classes. In the main class, it has a type long static variable that stores a number and will be accessed by allthreads, the main class also starts up all the threads. There are two types of threads, the first type is adding one to the static variable for 100000 times and the other thread is subtracting one from the static variable for 100000 times. In the main class, it will start up two different types of threads at same time for certain amount of times. After all threads start, the main thread will wait for them to finish and then it will print out the final value of the number.

Result:

The expected value of the final number is zero because the adding thread adding a value to the static variable while the subtracting thread subtracting the same value from it. Therefore as long as the number of times adding threads run equal to the number of times that subtracting threads run, the final number should be zero.

However from the final value that is obtained from the program is not equal to the expected value, it is a random number every time the program runs. From the table below, it is shownthe result that is obtained from the program with five adding threads and five subtracting threads.

	Final values get from the program
1	-39140
2	1563
3	-11957
4	46929
5	-102709

Table 1: Final values get from the program

The final value and expected value are not the same number, this is proves that the thread collision has occurred.After one thread did all the calculation and is ready to the write that variable back, the original variable is already overwrote by other thread, so value this thread going to write will be wrong as well. As the number of threads increases, chances for thread collision increases.

Analysis:

From the result from above, it shows that the threads had collided with each other. This proves that when the threads are not thread-safe, thread collision can occur.