**Question-1:**

False, the given code snippet has more than 1 possible outcome. Three possible outcomes with the given code are as follows:

Possibility 1:

Starting1

Exiting1

Starting2

Exiting2

Possibility 2:

Starting2

Exiting2

Starting1

Exiting1

Possibility 3:

Starting1

Starting2

Exiting2

Exiting1

* As the outputs are unbuffered system calls (printf() statements), it is not guaranteed that the calls from thread 1 or thread 2 will get executed in order. For example, when printf(“\nStarting2”) from thread 2 gets executed, the next statement that gets executed because of how the OS handles the system calls can be either printf(“\nExiting2”) or printf(“\nStarting1”).

**Question-2:**

False, the given code snippet has more than one possible output. As the order of execution of the thread functions is non-deterministic (handled entirely by the OS) and each thread has its own program stack (hence they won’t overwrite each other’s actual parameters in the called function), the output of the given code snippet can be any of the followings:

Possibility 1:

5

12

Possibility 2:

12

5

**Question-3:**

False, the given code snippet has more than one output. This code snippet has a race condition as both of the threads are trying to access and modify a common ‘sum’ variable inside the worker function which is not protected by any mutex/lock. As a result, the output of this program can be any of the following:

Possibility 1:

Sum: 8

Possibility 2:

Sum: 3

Possibility 3:

Sum: 5

**Question-4:**

False, this code snippet has more than one output. As the critical section of the worker function is protected by mutex locks, depending on the order of thread execution, the output of this program will be one of the following:

Possibility 1:

2

5

Possibility 2:

3

5

**Question-5:**

Even though this given code snippet is using mutex locking, the critical section in the worker function is not thread-safe as we are using two different mutex objects. As a result, when both threads request to lock their own mutex objects, both can get access to the critical section at the same time. This would result in a race condition. The output of this program can be any of the following:

Possibility 1:

2

5

Possibility 2:

3

5

Possibility 3:

2

2

Possibility 4:

3

3