```
breaklines=true,
frame=single,
language=C++,
keywordstyle=blue,
commentstyle=green!50!black,
stringstyle=red
```

Introduction This report documents the implementations and analyses for the thread-safe data structures lab assign
Exercise 1: Thread-Safe Queue and Stack [language=C++]thread$_s afe_d ata_s tructures.cpp$
Explanation: Describe how mutexes ensure thread safety in ThreadSafeQueue and ThreadSafeStack. Explain the
Analysis: Discuss thread synchronization behavior and consumer termination logic.
Screenshot: Include a screenshot of compiling and running thread$_s afe_d ata_s tructures.cpp.[h][width =]1_s creenshot$
Exercise 3: Thread-Safe Priority Queue [language=C++]priority$_q ueue.cpp$
Explanation: Describe how the priority queue maintains order and ensures thread safety.
Analysis: Analyze the behavior of concurrent pushes and pops.
Screenshot: Include a screenshot of compiling and running priority$_q ueue.cpp.[h][width =]2_s creenshot.pngCompil$
Exercise 4: Thread-Safe Circular Buffer [language=C++]circular$_b uffer.cpp$
Explanation: Explain the use of condition variables for synchronization.
Analysis: Compare to ThreadSafeQueue from Exercise 1.
Screenshot: Include a screenshot of compiling and running circular$_b uffer.cpp.[h][width =]3_s creenshot.pngCompi$
Exercise 5: Thread-Safe Deque [language=C++]thread$_s afe_d eque.cpp$
Explanation: Describe the challenges of synchronizing access to both ends.
Analysis: Analyze concurrent operations on the deque.
Screenshot: Include a screenshot of compiling and running thread$_s afe_d eque.cpp.[h][width =]4_s creenshot.pngCom$
Exercise 6: Thread-Safe Linked List [language=C++]thread$_s afe_l inked_l ist.cpp$
Explanation: Discuss maintaining list integrity during concurrent operations.
Analysis: Analyze thread safety and performance considerations.
Screenshot: Include a screenshot of compiling and running thread$_s afe_l inked_l ist.cpp.[h][width =]5_s creenshot.png$