Exercise 1. You are required to write a program to help you understand multi-thread and semaphore/mutex concepts. The program needs to create one global variable and two threads.

In thread 1, the program needs to sum up from 1 to 100, and outputs “thread one” information. In thread 2, the program needs to sum up from -100 to -1, and outputs “thread two” information.

The two threads need to run mutually. In more details of each thread inside, firstly, the program waits for a semaphore, secondly, the program is locked to only run this thread, thirdly, do the calculation, fourthly, unlock the program; fifthly, send a semaphore to another thread.

Also, the thread 2 runs firstly and send a semaphore to thread 1 when thread 2 finishes. Then, thread 1 begin to run until it finishes all its work. When the program finishes, it needs to release all the resource and exit correctly.

A program framework is given, you need to fill the program and satisfy the requirement.

Exercise 2. You are required to write a program to help you understand pipe concept. The program needs to use fork to separate the program into two sub-processes. In the parent process, the program writes a content to pipe. In the child process, the program reads the content from pipe and outputs it to the monitor.

A program framework is given, you need to fill the program and satisfy the requirement.