Lab 2 Report

Exercise 1:

1. Complete the program by adding the missing signal and wait code in the two routines, sender() and receiver(). Compile and run the program

   Done. Attached code changes in the zip file

2. Switch the order of the following two statements in main():
   pthread_create(&tid2, NULL, (void *)receiver, NULL);
   pthread_create(&tid1, NULL, (void *)sender, NULL);
   Compile and run the modified program. What happens? Can you explain?

   The receiver does not even go into waiting state and the sender sends the signal.

3. Add a second receiver to the program by creating a third thread in main() to run the receiver() routine:
   pthread_create(&tid3, NULL, (void *)receiver, NULL);
   ...
   pthread_join(tid3, NULL);
   Don't change the sender() and receiver() routines themselves. Compile and run the modified program.
   What happens? If it doesn't work, fine a simple way to fix it

   The receiver that is in the waiting state is never woken up. This causes the output to Freeze. This can be fixed by adding the broadcast signal.

Exercise 2:

1. What do you expect the program to print? Compile and run the program. Does it produce the expected result? Can you explain the program's output?

   Yes. The output is as expected. There are 3 threads created and each thread calculates the array value at an index based on the results of the other threads. Example: a[2] in thread 2 is calculated using the result of thread 0 where a[0] is calculated.

2. Insert barrier synchronization into the program, so that each of the statements in the worker() routine is guaranteed to have completed across all worker threads

before the next statement starts. Compile and run the program. Does the output meets your expectation?

Observation 1: When I initialize the barrier in the main function with 4 threads (3 child and 1 main) and then add barrier wait in the worker routine, the output hangs because the barrier is always waiting for the 4th main thread but that thread never enters the worker routine.
If the barrier is initialized for only 3 child threads and if barrier wait is added in the worker thread, then the output is as expected since the barrier is lifted once all the 3 child threads arrives at the barrier.

Observation 2: If barrier wait is added at each step then the output changes to 1,2,0.

3. Replace the pthread_join() loop with a barrier synchronization. Compile and run the modified program

   Done

Exercise 3:

1. Compile this program, and run it multiple times. Do you observe any problems? Try to explain the results

   Yes. The buffer size is going out of bound. Here the consumer is continuing to remove the data even when the buffer is empty and producer has not added any data into the buffer. The index is going out of range

2. Now insert a simple "busy-waiting" synchronization into the producer() routine:
   while (idx == BUFSIZE)
   ; // busy waiting
   and a similar one into the consumer() routine:
   while (idx == 0)
   ; // busy waiting
   Compile and run the modified program. Does the program work now? Do you observe any instance where the array index value is out of the buffer bounds?

   It does not work. There are instances where the array index value is out of buffer bounds

3. Replace the busy-waiting synchronization with signal-and-wait synchronization using condition variables

Done

From this lab assignment, I got more perspective on thread synchronization, using condition variables, signal and wait, barrier wait.