Assignment 3

- Description of your code and the logical and implementation details. –

  The program makes threads for all the readers and writers who have access to the data set provided that no writer is editing it at that moment.
  Writers can edit data at the instances when no readers have access to it.

  Main function is solely for user input, declaring semaphores and, creation and joining of threads.

```
void *reader(void *arg)
```

  This function checks semaphore mutex to see if the previous reader is done reading so that it can gain access to the data.
  Then it checks semaphore wrt to see if a writer has access to the data set and if i does, the reader thread waits for the writing to be completed.
  Once it's done reading, it unlocks the semaphore mutex so other readers waiting to access data can enter.
  Once all readers are done, wrt is unlocked, so writer threads can have access to the data.

```
void *writer(void *arg)
```

  This function checks semaphore wrt to make sure no reader or writer thread has access to the data while it is making changes.
  It performs the write operation which is writing the writer number into the queue.
  Then it unlocks wrt so that other reader/writer threads can access it.

- Description of how to compile and test the program –

  gcc CAOS.c -pthread
  ./a.out

  OR run the makefile
  make
  ./output

- The inputs the user should give. –
  Number of readers
  Number of writers

- Expected output (and how to interpret it) –
  Writers add their writer numbers to the queue when they're reading it. When readers are reading, they print the whole data set they're reading which is essentially the writer numbers of the writers that accessed the data set before the reader read it.

- Error-values and how to interpret them –

```
"Scanf error: Only numbers expected"
"Scanf error: There should be at least 1 reader"
```