Homework #1: Producer / Consumer (100 points)
Submit a compressed (.tgz) file with **source code** and **Makefile** to Canvas

You must write a *multithreaded* **producer / consumer** program in **C++** that:

- Takes *four* command line arguments:
  - `UNIX> ./hw1 num_prod num_cons buf_size num_items`
    - *`num_prod`* := number of producer threads (> 0)
    - *`num_cons`* := number of consumer threads (> 0)
    - *`buf_size`* := maximum size of buffer (> 0)
    - *`num_items`* := *total* number of items to *produce* (> 0)

- **Producer** threads:
  - Produce items and put them in a shared global buffer
    - Buffer can hold at most *`buf_size`* items
  - Each item must contain *two* numbers:
    - An identification number (from 0 to *num_items-1*)
    - A random sleep time (between 200 - 900 μsecs)
  - *Before* producing each item, producer must *sleep* for random amount of time between 300 - 700 μsec
    - Simulates work to *produce* item
  - If buffer is full, *producer* must *wait* until there's available space
  - *`num_items`* must be evenly divided between *`num_prod`* producer threads
    - E.g., if *`num_items`* = 1000 and *`num_prod`* = 10, then each producer thread will produce 100 *unique* items
      - Producer 0 will produce items 0-99
      - Producer 1 will produce items 100-199
      - etc.

- **Consumer** threads:
  - Consume items from shared global buffer
  - On consumption, consumer will:
    - *Sleep* for amount of time specified in item being consumed
    - Print the consumer number and item identification number to *stdout*
      - Consumer number is between *0 and `num_cons` - 1*
  - If no items in buffer, consumer must *wait* for items

- After all *producer* threads have finished, you must print a message to ***stderr***
  - Greatly simplifies program termination
  - User terminates program w/ CNTL-C after *all* items have been *consumed*

- **Your solution must NOT result in deadlock.**
  - You must use appropriate thread synchronization

- **You must support *any number* of producers, consumers, and buffer sizes**
  - Limited only by system resources (e.g., max number of threads)

Hints:

- usleep()
- mutex
- semaphore
- pthreads
- C++ standard template library
- Work incrementally and print *useful* debugging messages

Examples:

```
UNIX> ./hw1
usage: ./hw1 num_prod num_cons buf_size num_items
```

//one producer, one consumer, buffer size = 10, 5 items

```
UNIX> ./hw1 1 1 10 5
  0: consuming     0
  0: consuming     1
  0: consuming     2
  0: consuming     3
  0: consuming     4
DONE PRODUCING!!
<CNTL-C>
```

//10 producers, 100 consumers, buffer size = 1, 100 items

```
UNIX> ./hw1 10 100 1 100 > output.txt
DONE PRODUCING!!
<CNTL-C>
```

//verify that 100 items were consumed

```
UNIX> wc -l output.txt
100 output.txt

UNIX> head -n 3 output.txt
  0: consuming     0
  1: consuming     1
 87: consuming     2

UNIX> tail -n 3 output.txt
 95: consuming    97
 96: consuming    98
 97: consuming    99
```

//100 producers, 10 consumers, buffer size = 10, 10000 items

```
UNIX> ./hw1 100 10 10 10000 > output.txt
DONE PRODUCING!!
```

//… wait a few seconds… (give consumers time to finish)

```
<CTRL-C>
```

//verify that 10000 items were consumed

```
UNIX> wc –l output.txt
 10000  output.txt

UNIX> head -n 3 output.txt
  0: consuming     0
  1: consuming     1
  5: consuming     5

UNIX> tail -n 3 output.txt
  6: consuming  9997
  0: consuming  9998
  9: consuming  9999
```

//200 producers, 100 consumers, buffer size = 1000, 1000000 items

```
UNIX> ./hw1 200 100 1000 1000000 > output.txt
DONE PRODUCING!!
```

//… wait a few seconds… (give consumers time to finish)

```
<CTRL-C>
```

//verify that 1000000 items were consumed

```
UNIX> wc -l output.txt
1000000 output.txt

UNIX> head -n 3 output.txt
  0: consuming     0
  1: consuming     1
  2: consuming     2

UNIX> tail -n 3 output.txt
 89: consuming 999997
 99: consuming 999998
 99: consuming 999999

UNIX>
```