


# prodcon

**Due** Apr 21 by 11:59pm      **Points** 100

in this assignment you will be writing a multi-threaded producer/consumer engine. your engine will be supplied shared libraries with producer/consumer logic that your engine will run. your engine will be invoked as follows:

```
./prodcon shared_lib consumer_count producer_count optional_args ...
```

for example, word count is the "hello world" of parallel processing. word count simply counts all the words in a file. it outputs a list of words in the file with the number of times that word appeared. you are provided an example implementation of [wordcount.c](#) , which you can compile using:

```
cc -fPIC -shared wordcount.c -o libwordcount.so -lpthread
```


this will compile and link a shared library called libwordcount.so. we can then run using

```
./prodcon ./libwordcount.so 2 2 words
```


make sure you put the path to libwordcount.so. you must have a /. **./prodcon** will load the **libwordcount.so** library and use the **run\_producer** function provided in that library to generate a list of words to pass to producers. the **assign\_consumer** function in **libwordcount.so** will indicate which consumer a given word should be routed to. the **run\_consumer** function will consume the words and accumulate counts which it will print once all the words have been consumed. for example, if we use [words](#) to run the above command, we get:


```
list 1
five 5
this 1
one 1
four 1
two 2
words 1
a 1
three 1
is 1
big 1
of 1
```

you will need to startup a thread for each producer and consumer. they will all run concurrently until all the producers have stopped producing input and all the consumers have finished consuming everything.

to get you going, i'm providing you will an example [prodcon.c](#)  implementation. IT IS NOT CORRECT. IT IS SINGLE THREADED. it does show you how to load libraries and get function pointers from them. feel free

to use the code in your implementation.

i'm also providing you another shared library for you to test with: [randnums.c](#) . it generates random numbers and sums them up.

you will submit a zip file with exactly one file in it: prodcon.c. use the [prodcon.h](#)  file i've given you to get the correct function prototypes for the plugin interface.

## rubric

one file named prodcon.c in zipfile	5
one thread is started for each producer and consumer	10
consumers do not finish until all output from producers have been consumed	10
the assign_consumer function is used correctly to assign output	10
producers and consumers all run concurrently	10
logic is correct and readable	20
optional command line arguments are passed correctly to functions	5
program builds cleanly without errors or warnings	10
all memory is freed up (no memory leaks)	10
no limits on the number of producers or consumers	10

