CSC112 Lab1: Multi-Threaded Programming

You are given a C program "pthread vows cons.c" that creates two threads (`vow` and `cons`) that print words starting with vowels and consonants, respectively, while maintaining the original sequence. It uses the syscall `sched_yield()` to allow threads to take turns.
Compile the program using command line (or VS Studio Code IDE):

```
gcc -pthread -o threadtest threadtest.c
```

Run the program by supplying a list of words as arguments, for example:

```
./threadtest apple banana orange grape kiwi umbrella
```

The output looks like this:

```
Vowel: apple
Consonant: banana
Vowel: orange
Consonant: grape
Consonant: kiwi
Vowel: umbrella
```

The program works as follows:
1. Input Handling: The program takes command-line arguments as input. Each argument is treated as a word.
2. Thread Creation: The `vow` thread prints words starting with vowels. The `cons` thread prints words starting with consonants.
3. Turn-Based Synchronization:
- A shared variable `turn` determines which thread should process the current word:
- `turn == 0`: Vowel thread's turn.
- `turn == 1`: Consonant thread's turn.
- Each thread checks whether it's its turn and processes the word accordingly. If it's not its turn, it calls `sched_yield()` to yield control to the other thread.
4. Word Processing:
- Each thread checks whether the current word starts with a vowel or consonant and prints it accordingly.
- The `current_index` ensures that both threads process words in sequence.

The system call `sched_yield()` lets the calling thread voluntarily give up control to the OS scheduler, in order to give other active threads a chance to run, The two threads use `sched_yield()` to take turns in printing out the vows and cons. The program preserves the original word sequence in the argument list because both threads operate on a shared index `current_index` and take turns based on the `turn` variable. No mutexes or explicit locking mechanisms are used.

Your job is to modify the program to use the mutex API instead of `sched_yield()`, i.e.,

```
pthread_mutex_t lock;
pthread_mutex_lock(&lock);
pthread_mutex_unlock(&lock);
```

Hints:
1. You still need the shared variable `current_index`, but not the shared variable `turn`.

2. You need to call `pthread_mutex_lock(&lock)` and `pthread_mutex_unlock(&lock)` within functions `print_vowels() and print_consonants()` to ensure mutual exclusion between them, since they are not yielding to each other voluntarily.

Please submit the following on Canvas:

1. "pthread vows cons mutex.c", and screenshots for running it against some inputs. If your program output does not preserve the original word sequence, i.e., it prints out all the vows before printing out all the cons, then you will not get any points.
2. A short PDF report explaining the code you have written and the execution results.