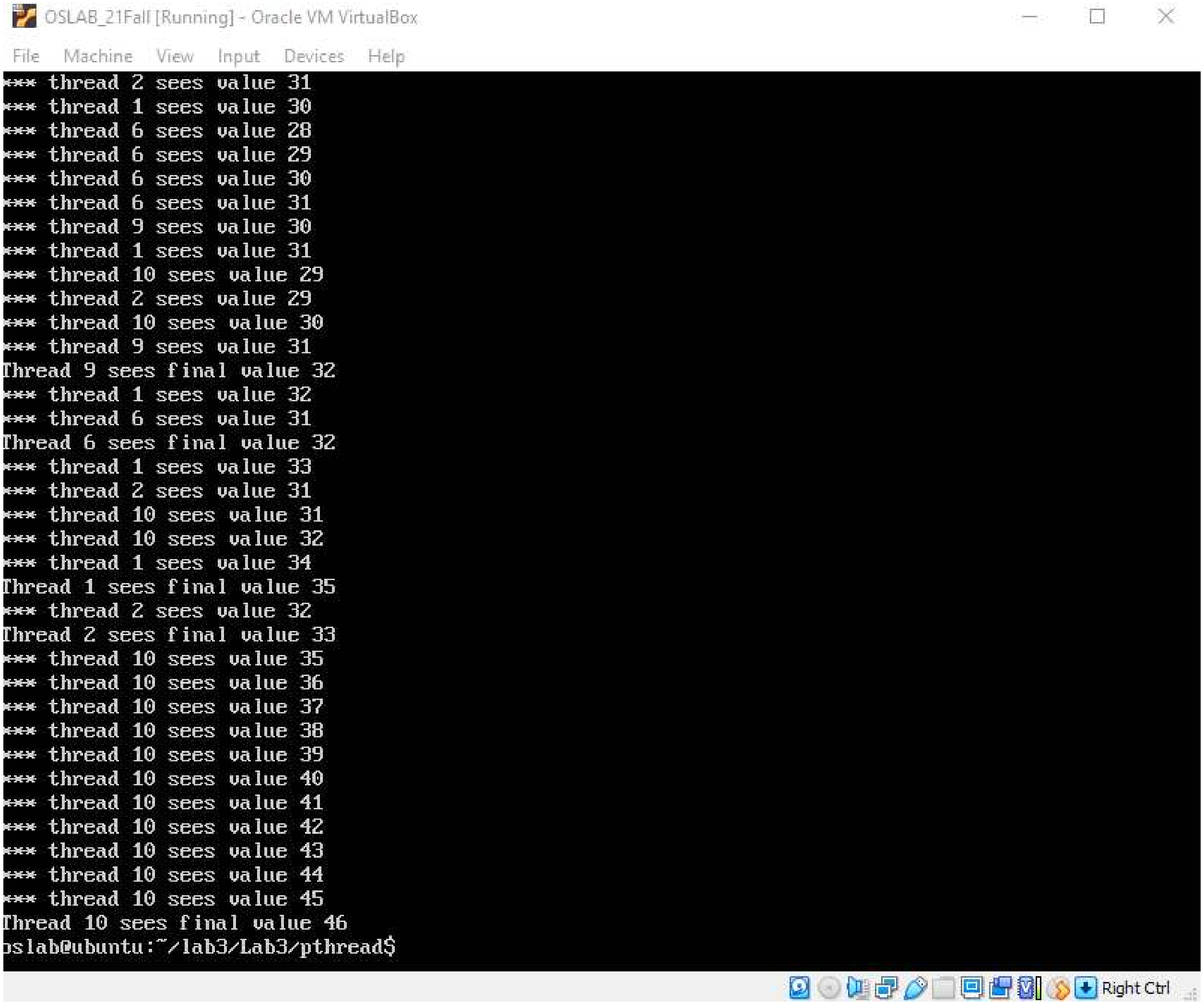
Jorge Leon

6095718

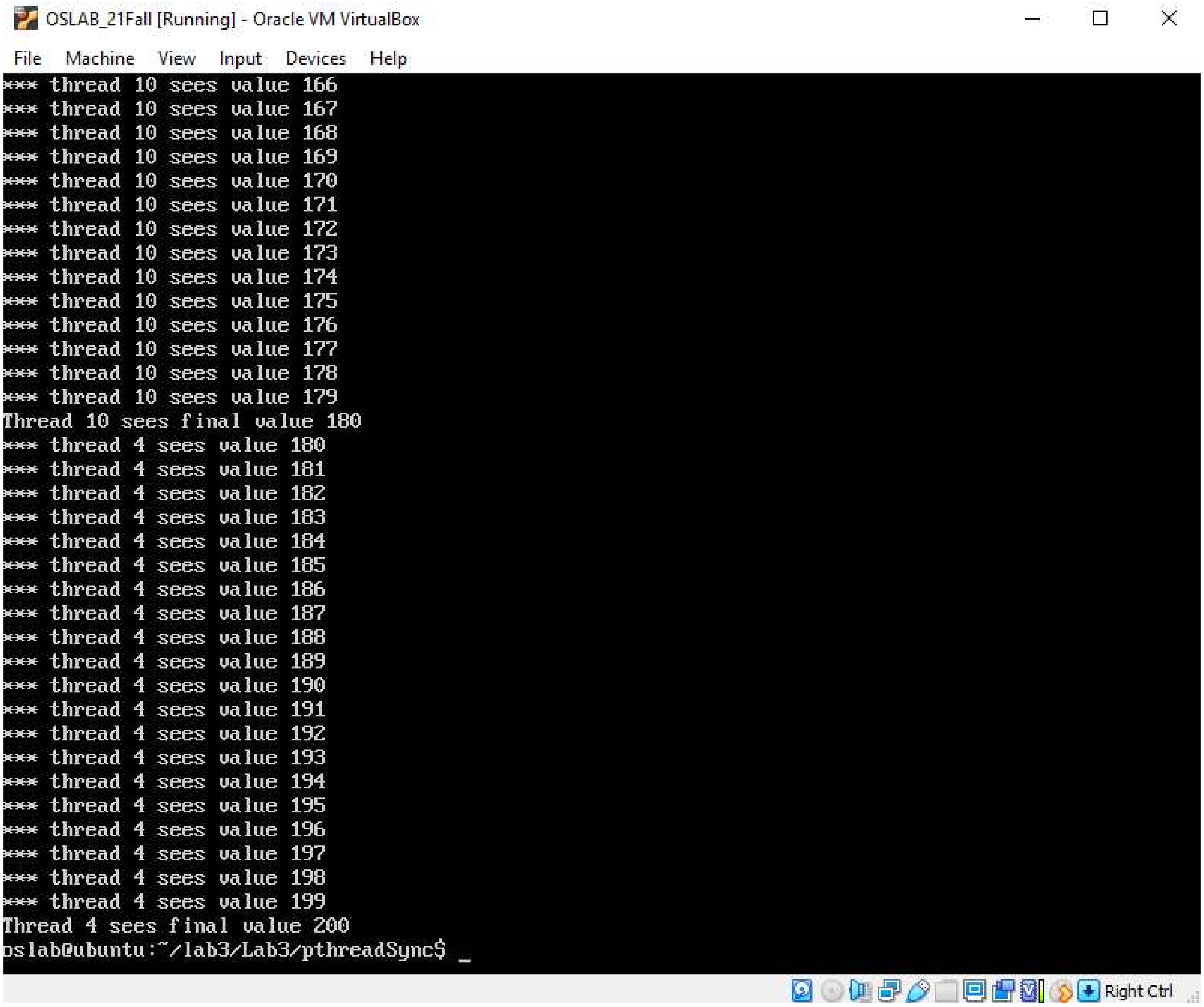
PART 1



```
*** thread 2 sees value 31
*** thread 1 sees value 30
*** thread 6 sees value 28
*** thread 6 sees value 29
*** thread 6 sees value 30
*** thread 6 sees value 31
*** thread 9 sees value 30
*** thread 1 sees value 31
*** thread 10 sees value 29
*** thread 2 sees value 29
*** thread 10 sees value 30
*** thread 9 sees value 31
Thread 9 sees final value 32
*** thread 1 sees value 32
*** thread 6 sees value 31
Thread 6 sees final value 32
*** thread 1 sees value 33
*** thread 2 sees value 31
*** thread 10 sees value 31
*** thread 10 sees value 32
*** thread 1 sees value 34
Thread 1 sees final value 35
*** thread 2 sees value 32
Thread 2 sees final value 33
*** thread 10 sees value 35
*** thread 10 sees value 36
*** thread 10 sees value 37
*** thread 10 sees value 38
*** thread 10 sees value 39
*** thread 10 sees value 40
*** thread 10 sees value 41
*** thread 10 sees value 42
*** thread 10 sees value 43
*** thread 10 sees value 44
*** thread 10 sees value 45
Thread 10 sees final value 46
oslab@ubuntu:~/lab3/Lab3/pthread$
```

The unsychranised version will sometimes have a thread working on the shared value but before it can modify it will switch to a new thread. After the it's the thread's time again, it will then modify it. But this modification is an old value so it basically nullify anything that happen

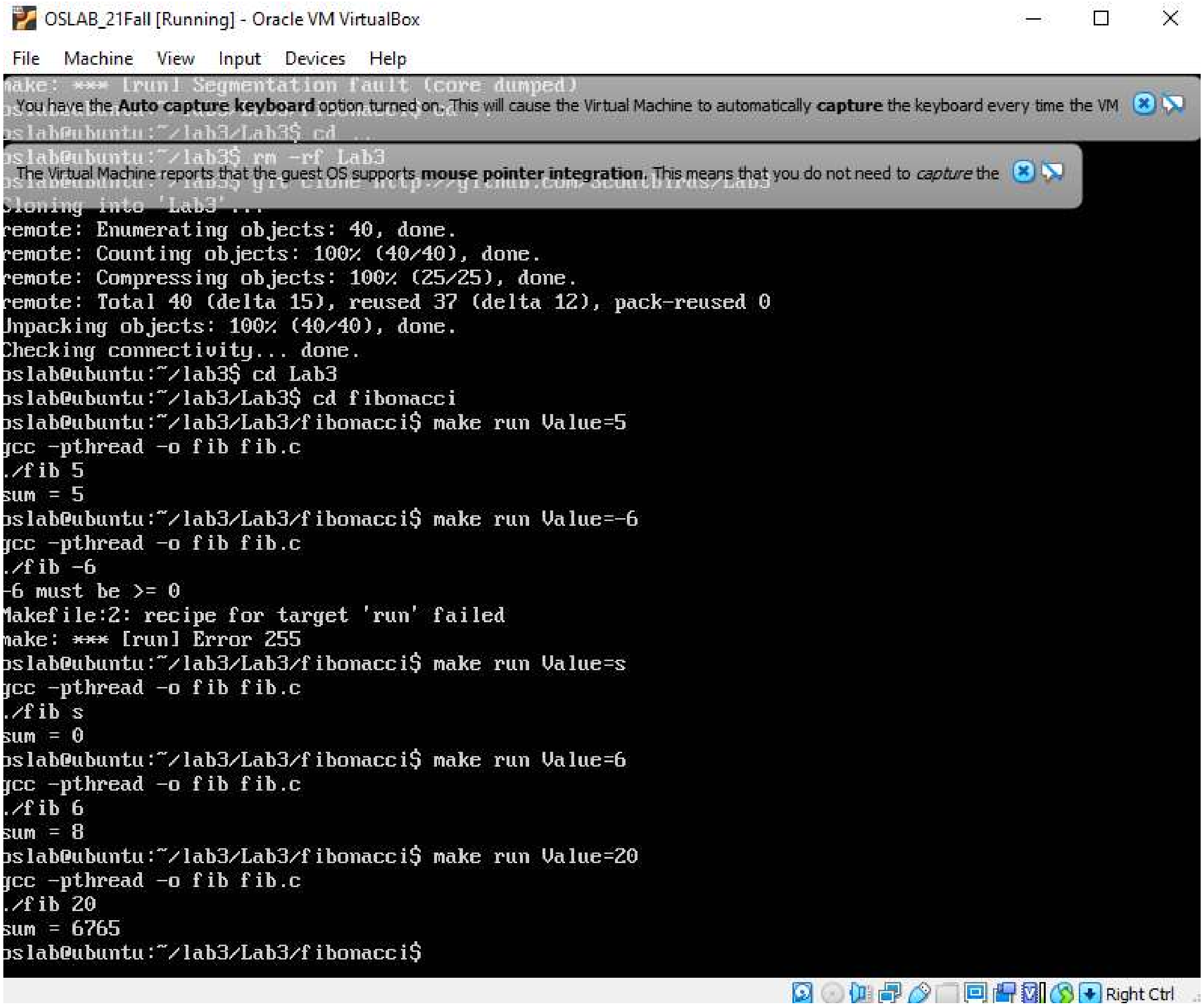
PART 2



```
*** thread 10 sees value 166
*** thread 10 sees value 167
*** thread 10 sees value 168
*** thread 10 sees value 169
*** thread 10 sees value 170
*** thread 10 sees value 171
*** thread 10 sees value 172
*** thread 10 sees value 173
*** thread 10 sees value 174
*** thread 10 sees value 175
*** thread 10 sees value 176
*** thread 10 sees value 177
*** thread 10 sees value 178
*** thread 10 sees value 179
Thread 10 sees final value 180
*** thread 4 sees value 180
*** thread 4 sees value 181
*** thread 4 sees value 182
*** thread 4 sees value 183
*** thread 4 sees value 184
*** thread 4 sees value 185
*** thread 4 sees value 186
*** thread 4 sees value 187
*** thread 4 sees value 188
*** thread 4 sees value 189
*** thread 4 sees value 190
*** thread 4 sees value 191
*** thread 4 sees value 192
*** thread 4 sees value 193
*** thread 4 sees value 194
*** thread 4 sees value 195
*** thread 4 sees value 196
*** thread 4 sees value 197
*** thread 4 sees value 198
*** thread 4 sees value 199
Thread 4 sees final value 200
oslab@ubuntu:~/lab3/Lab3/pthreadSync$ _
```

For the synchronized version of the code, the threads are not allowed to modify the shared value until one thread is done because of the locking function.

Fibonacci



You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture** the keyboard every time the VM

The Virtual Machine reports that the guest OS supports **mouse pointer integration.** This means that you do not need to *capture* the

```
make: *** [run] Segmentation fault (core dumped)
oslab@ubuntu:~/lab3/Lab3/fibonacci$ cd ..
oslab@ubuntu:~/lab3/Lab3$ cd ..
oslab@ubuntu:~/lab3$ rm -rf Lab3
oslab@ubuntu:~/lab3$ git clone http://github.com/scoutbirds/Lab3
Cloning into 'Lab3'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 40 (delta 15), reused 37 (delta 12), pack-reused 0
Unpacking objects: 100% (40/40), done.
Checking connectivity... done.
oslab@ubuntu:~/lab3$ cd Lab3
oslab@ubuntu:~/lab3/Lab3$ cd fibonacci
oslab@ubuntu:~/lab3/Lab3/fibonacci$ make run Value=5
gcc -pthread -o fib fib.c
./fib 5
sum = 5
oslab@ubuntu:~/lab3/Lab3/fibonacci$ make run Value=-6
gcc -pthread -o fib fib.c
./fib -6
-6 must be >= 0
Makefile:2: recipe for target 'run' failed
make: *** [run] Error 255
oslab@ubuntu:~/lab3/Lab3/fibonacci$ make run Value=s
gcc -pthread -o fib fib.c
./fib s
sum = 0
oslab@ubuntu:~/lab3/Lab3/fibonacci$ make run Value=6
gcc -pthread -o fib fib.c
./fib 6
sum = 8
oslab@ubuntu:~/lab3/Lab3/fibonacci$ make run Value=20
gcc -pthread -o fib fib.c
./fib 20
sum = 6765
oslab@ubuntu:~/lab3/Lab3/fibonacci$
```

In this program when a thread is created and works on the Fibonacci sequence, a new thread will have to wait until the old thread is finished.