UID: 3035488160

COMP3230 – Tutorial 3 Exercise 3

```
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffff77c4700 (LWP 11494)]
[New Thread 0x7ffff6fc3700 (LWP 11495)]
[Switching to Thread 0x7ffff77c4700 (LWP 11494)]

Thread 2 "hw3" hit Breakpoint 1, get_instance () at hw3.c:24
24        return ctx;
(gdb) print ctx->initialized
$1 = false
(gdb) continue
Continuing.
[Switching to Thread 0x7ffff6fc3700 (LWP 11495)]

Thread 3 "hw3" hit Breakpoint 1, get_instance () at hw3.c:24
24        return ctx;
(gdb) print ctx->initialized
$2 = false
(gdb) continue
Continuing.
name=A  id=1
name=A  id=1
[Thread 0x7ffff77c4700 (LWP 11494) exited]
[Thread 0x7ffff6fc3700 (LWP 11495) exited]
[Inferior 1 (process 11490) exited normally]
(gdb)
```

There is racing condition between two threads. Both threads get_instance in the breakpoint 24 as they access the critical part without mutex.

```
chngai@workbench Desktop> ./hw3
name=A  id=1
name=A  id=1
chngai@workbench Desktop>
```

Solution in hw3. Use pthread_mutex_lock and pthread_mutex_unlock to prevent both threads access the critical part at the same time.