Optional Assignment – 1
Locks

Anikait Sahota (2018016)

```
struct thread {
	void *esp ;
	struct thread *next;
	struct thread *prev;
	void* stack_base ;
};
```

```
int WAIT_LIST_FLAG = 0 ;
struct thread *ready_list = NULL;     // ready list
struct thread *cur_thread = NULL;     // current thread
int EXIT_FLAG = 0 ;
struct thread *exit_thrd = NULL ;
```

```
void sleep(struct lock *lock)
{
	// printf("%p %p %p\n",wait_lists , &(lock->wait_list) , lock->wait_list );
	push_back((struct thread** )(&(lock->wait_list)) , cur_thread ) ;
	WAIT_LIST_FLAG ++ ;

	// struct thread** t = (void*) wait_lists ;
	// printf("%p %p %p\n",*t , (*(struct thread**)wait_lists) , lock->wait_list );
	// printf("sleep\n" );
	schedule() ;
	// printf("%p %p\n",lock->wait_list ,(struct thread*)lock->wait_list );
}
```

```
void wakeup(struct lock *lock)
{
	struct thread* thrd = pop_front((struct thread** )&(lock->wait_list)) ;
	if(thrd != NULL)
		WAIT_LIST_FLAG-- ;
	push_back(&ready_list , thrd) ;
	// printf("wakeup\n" );
	// schedule() ;
}
```

```
void foo(void *ptr)
{
	struct lock *l = (struct lock*)ptr;
	acquire(l) ;
	int val;
	val = counter;
	val++;

	// release(l) ;
	thread_yield() ;
	// acquire(l) ;

	counter = val;
```

```
        release(l) ;        // this is above thread_exit because a thread can have multiple critical section to execute
        thread_exit();
}
```

```
main thread exiting : counter:1024000
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 5304
        Average resident set size (kbytes): 0
```

Yes race2 causes deadlock.

The strategy I had dicussed in the assignment-2 was to free the thread's stack and thread in thread_exit(). But since thread is needed till the context switch and after that the thread will never be executed. So we need to free the thread in next thread's execution cycle.

SO my approach in assignment 2 was correct, however implemntation requires more depth.

Finished...