Mico Raphael F. Cuarto
CS 3202

1.
Busy Waiting
```
void* thread_busy_waiting(void* rank) {
   long thread_id = (long) rank;
   int local_n = n / NUM_THREADS;
   double local_a = a + thread_id * local_n * h;
   double local_b = local_a + local_n * h;
   double my_sum = trapezoid_area(local_a, local_b, local_n);

   while (flag != thread_id); // busy-wait
   global_sum += my_sum;
   flag++;
   return nullptr;
}
```
Mutex
```
void* thread_mutex(void* rank) {
   long thread_id = (long) rank;
   int local_n = n / NUM_THREADS;
   double local_a = a + thread_id * local_n * h;
   double local_b = local_a + local_n * h;
   double my_sum = trapezoid_area(local_a, local_b, local_n);

   pthread_mutex_lock(&mutex);
   global_sum += my_sum;
   pthread_mutex_unlock(&mutex);
   return nullptr;
}
```
Semaphore
```
void* thread_semaphore(void* rank) {
   long thread_id = (long) rank;
   int local_n = n / NUM_THREADS;
   double local_a = a + thread_id * local_n * h;
   double local_b = local_a + local_n * h;
   double my_sum = trapezoid_area(local_a, local_b, local_n);

   sem_wait(&sem);
   global_sum += my_sum;
   sem_post(&sem);
   return nullptr;
}
```

2.

Method	Advantages	Disadvantages
Busy-wait	Easy to implement	Wastes CPU time, inefficient
Mutex	Efficient, cleaner mutual exclusion	Risk of deadlock if misused
Semaphore	Good for signaling, flexible	Slightly more complex, less intuitive