**Assignment 5 – Futures Part 2**

In this assignment, we implemented 3 types of Futures,

1. FUTURE EXCLUSIVE
2. FUTURE SHARED
3. FUTURE QUEUE
4. FUTURE EXCLUSIVE:
   1. In this part, we implemented a future which would have a one to one relationship between a thread calling future\_get() and future\_set(). Initially, the future state will be FUTURE\_EMPTY. When a thread calls future\_set(), it will set the value passed as a argument into the function and any further calls to future\_set() will fail. The same goes for future\_get(). Initially, future\_get() will take the value set by the future\_set() and all other calls to future\_get() function will fail.
5. FUTURE SHARED:
   1. In this part, we implemented a future which would have one to many relationship between future\_set() and future\_get(). Only one thread can call future\_set() and all other calls to future\_set() throws and error. However, many threads can call future\_get(). If a future doesn’t have a value, we enqueue the thread in a get\_queue of the future. As soon as a future is set, all threads, waiting on the value will be resumed.
6. FUTURE QUEUE:
   1. In this part, we implemented a future which would have a many to many relationship between future\_set() and future\_get(). Multiple threads can call future\_set() and future\_get(). If a thread calls future\_set() and threads are waiting for value in get\_queue then the thread calling future\_set() sets the value and resumes only one thread from get\_queue based on first come first serve basis. If there is no thread waiting in get\_queue then thread calling future\_set() enqueues itself in set\_queue. Similarly, if a thread calls future\_get and there are threads waiting to set value in set\_queue the calling thread dequeues only one thread from from set\_queue and gets the value released from set\_queue. If there is no thread waiting in set\_queue and should get the value set by the thread released from set\_queue. If there is no thread waiting in set\_queue to set value, thread calling future\_get enqueues itself in get\_queue.

To implement get\_queue and set\_queue I created a new struct in futures.h file which has the definition of the queue structure (called qent). This queue structure contains value, pid and next pointer of the struct. Thisa value, is used in FUTURE\_QUEUE.

Fibonacci:

In this program, I created a new array of futures (called fibfut) which holds N number of futures (N is an argument passed by the user). Then, I allocate a future shared\_queue to each array element inside the array. I then resume the threads and pass the initial number of the Fibonacci series (1).