CS460_Life

- "Your code must contain a comment explaining how you will divide up the work among the N threads. The plurality of points for M1 will be given to the comment."
- How will you divide the work among the threads?
- What data will be shared by multiple threads?
- What data will need to be synchronized/protected across threads?
- What happens to each thread at the end of a generation?
- Clarification: 4 threads means you can have 4 threads and main() (really 5 threads) active at any one time.

CondWaitThread.c

```
// all threads share one struct

typedef struct ThreadArgs
{
    // data, mutex, and cond to signal a thread to start
    int startFlag ;
    pthread_mutex_t sStartMutex;
    pthread_cond_t sStartWaitCond;

    // data, mutex, and cond to signal that all threads have finished

    int finishedFlag ; // initialize to zero
    pthread_mutex_t sFinishedMutex;
    pthread_cond_t sFinishedWaitCond;

} ThreadArgs;
```

Thread One - the waiting thread -- wait for 2 other threads to finish

```
pthread_mutex_lock(&sThreadArg.sFinishedMutex);


while(2 != sThreadArg.finishedFlag)
{

  // only one thread should call pthread_cond_wait
  // per pthread_cond_t

  pthread_cond_wait(&sThreadArg.sFinishedWaitCond,
                    &sThreadArg.sFinishedMutex);

}


sThreadArg.finishedFlag = 0;

pthread_mutex_unlock(&sThreadArg.sFinishedMutex);
```

Thread Two and Three - the finishing threads

```
// psThreadArg is a pointer to sThreadArg

pthread_mutex_lock(&psThreadArg->sFinishedMutex);

psThreadArg->finishedFlag ++;

pthread_mutex_unlock(&psThreadArg->sFinishedMutex);

// many threads may call pthread_cond_signal
// for each pthread_cond_t

pthread_cond_signal(&psThreadArg->sFinishedWaitCond);
```