requirement enginering   MPI Game of Life

Peer to pear seems the best method here (fewer communications)

Setup (each process independently) ① → find domain Splitting
→ find adjasent domains to communicate with → fixed boundary / → Periodic
② → initalise this processors domain

① set_domain();
② initalise_subdomain();

GoL iterations

→ use a second array C-adj to count adjasent alive cells ← later optimisation may include Sparcity analysis (ie. Skip areas with only zeros) ← could do by tracking areas with nothing alive

→ two possible implementations : if cell alive : add to adjasent cell count or add regardless wether alive or dead ← will test which is faster with test_time();

→ send 4 messages which are the edge strips (columns can be sent as an MPI datatype which skips elements to get the correct value; lends itself to a Class based structure)

→ use Isend, Irecv with waitall on the recieves

→ add these messages to the C-adj just as before

→ then if c.adj == 2 or 3 cell is alive; else dead

→ write to save buffer

→ Repeat

↑ can recycle my code from homework assignments

↑ keep data, methods and MPI datatype together

Code Structure

```
Class GOL_grid() ← each processor has 1 instance
  public:
      width, height = int, int
      neighbour IDS = { top, right, bottom, Left }   use -1 to indicate boundary
      life_grid = bool*
      adj_grid = int*
      of_stream = Save file

    void write_to_file ← may need some kind of buffer to
                         prevent writing bottleneck
    void update_life_grid ← used filled adj to update life grid then reset adj

    MPI datatypes Row, Col
    void create_datatypes
```

Future problem consideration

1. how to partition the domain
2. dealing with self edges on periodic domains
3. avoiding a save bottleneck (save buffer?)
4. reconstruction of results

working Scribbles