Logic.c:

* Int init\_battleground
  + Initialisiert eine Matrix der Größe size
  + Ruft reset\_battleground auf
  + Gibt im Erfolgsfall 1, im Fehlerfall OUT\_OF\_MEMORY zurück
* Int init\_stats
  + Initialisiert die Statistikspeicher je nachdem, welche Schiffe es gibt
  + Ruft hierzu auch set\_ships\_stats auf
  + Gibt im Erfolgsfall 1, im Fehlerfall OUT\_OF\_MEMORY zurück
* Void set\_ships\_stats
  + Wertet ships[] aus und extrahiert die Schiffstypen
* Void update\_stats
  + Erhöht die Stats je nach getroffener Schiffsklasse
* Void reset\_battleground
  + Setzt die Matrix auf den Anfangszustand (Wasser überall) zurück
* Int rand\_set\_ships
  + Setzt die Schiffe zufällig auf dem Spielfeld
  + Ruft hierzu set\_ship auf
* Int set\_ship
  + Überprüft die Position, an die das Schiff gesetzt werden soll und schreibt es in die Matrix, falls möglich
  + Gibt im Erfolgsfall 1, sonst 0 zurück
* Int ship\_mass\_validation
  + Prüft, ob die Anzahl an Schiffen auch in die vorliegende Matrix passt
  + Gibt im Erfolgsfall 1, sonst 0 zurück
* Int shoot
  + Schießt auf das Feld an den übergebenen Koordinaten
  + Gibt zurück welche Schiffsklasse getroffen wurde
* Int check\_downed
  + Überprüft, ob ein Schiff versenkt wurde oder nicht und setzt dieses Schiff im Bedarfsfall auf versenkt
  + Ruft hierzu check\_downed\_helper (Hilfsmethode, hier nicht aufgeführt) und downed\_maker auf
  + Gibt 1 oder 0 zurück
* Int downed\_maker
  + Setzt ein Schiff rekursiv auf (D)owned
* Int is\_end\_game
  + Prüft ob das Spiel beendet ist (sich also noch Ziffern in der Matrix befinden)
  + Gibt 1 oder 0 zurück
* Int compare
  + Vergleichsfunktion für qsort in main() Vgl. http://www.cplusplus.com/reference/cstdlib/qsort/