```
%Make the board. Make the shark and position the shark.
%Calls play and makeBoard
function [] = startShark(rows, cols)

%Recursive function that implements top-level algorithm.
%oldShark is a shark with the previous position.
%Calls: print, plotBoard, getClosestGoldfish, moveToGoldfish, eatGoldfish
function [] = play(board, shark, oldShark)

%Goldfish disappears from board, shark moves into former goldfish location.
%Calls:
function [newBoard newShark] = eatGoldfish(board, goldfishCoords, shark)

%Prints an ascii version of board at interpreter prompt, showing all
%simulated items.
function [] = print(board, shark)

%Plots board, showing all simulated items.
%Calls: plotSquare
function [] = plotBoard(board, shark, oldShark)

%Plots a single square on board.
%Calls: patch
function [] = plotSquare(row, col, color)

%Shark moves one square at a time, horizontally or vertically, until
%shark is adjacent to goldfish. Returns updated copy of shark.
%Calls: isAdjacent, moveOneSquare, print, plotBoard
function newShark = moveToGoldfish(board, goldfishCoords, shark)

%Move shark one square closer to coords. New shark location must be a
%legal, unoccupied square.
%Calls: isLegalMove
function [canMove newShark] = moveOneSquare(board, coords, shark)

%Returns true iff board is empty at proposed position and position is
%valid for this board.
%Calls:
function legal = isLegalMove(board, proposedMove)

%Returns true iff coords and shark position are vertically or
%horizontally adjacent.
%Calls:
function flag = isAdjacent(coords, shark)
```

```
%For every element of the board matrix, if the element of
%the matrix is GOLDFISH, then calculate a distance. Keep track of
%the minimum distance so far AND the coords of the min distance. If no
%goldfish is found, isGoldfish is false. Otherwise, coords holds the row and col
%coordinates of the min distance goldfish.
%Calls: distance
function [isGoldfish coords] = getClosestGoldfish(board, shark)

%Takes the row, column coordinates of two points on the board and returns
%the distance between them.
%Calls:
function out = distance(x1, y1, x2, y2)

%Makes a matrix of the dimension parameters. Randomly places goldfish in
%about five percent of the board (that is, each square on the board has a
%five percent chance of being goldfish).
%Calls:
function board = makeBoard(rows, cols)
```