```
# Sonar Treasure Hunt

import random
import sys
import math

def getNewBoard():
    # Create a new 60x15 board data structure.
    board = []
    for x in range(60): # The main list is a list of 60 lists.
        board.append([])
        for y in range(15): # Each list in the main list has 15 single-character strings.
            # Use different characters for the ocean to make it more readable.
            if random.randint(0, 1) == 0:
                board[x].append('~')
            else:
                board[x].append('`')
    return board

def drawBoard(board):
    # Draw the board data structure.
    tensDigitsLine = '    ' # Initial space for the numbers down the left side of the board
    for i in range(1, 6):
        tensDigitsLine += (' ' * 9) + str(i)

    # Print the numbers across the top of the board.
    print(tensDigitsLine)
    print('   ' + ('0123456789' * 6))
    print()

    # Print each of the 15 rows.
    for row in range(15):
        # Single-digit numbers need to be padded with an extra space.
        if row < 10:
            extraSpace = ' '
        else:
            extraSpace = ''

        # Create the string for this row on the board.
        boardRow = ''
        for column in range(60):
            boardRow += board[column][row]

        print('%s%s %s %s' % (extraSpace, row, boardRow, row))

    # Print the numbers across the bottom of the board.
    print()
    print('   ' + ('0123456789' * 6))
    print(tensDigitsLine)

def getRandomChests(numChests):
    # Create a list of chest data structures (two-item lists of x, y int coordinates).
    chests = []
    while len(chests) < numChests:
        newChest = [random.randint(0, 59), random.randint(0, 14)]
        if newChest not in chests: # Make sure a chest is not already here.
            chests.append(newChest)
    return chests

def isOnBoard(x, y):
    # Return True if the coordinates are on the board; otherwise, return False.
```

```
    return x >= 0 and x <= 59 and y >= 0 and y <= 14

def makeMove(board, chests, x, y):
    # Change the board data structure with a sonar device character. Remove treasure chests
from the chests list as they are found.
    # Return False if this is an invalid move.
    # Otherwise, return the string of the result of this move.
    smallestDistance = 100 # Any chest will be closer than 100.
    for cx, cy in chests:
        distance = math.sqrt((cx - x) * (cx - x) + (cy - y) * (cy - y))

        if distance < smallestDistance: # We want the closest treasure chest.
            smallestDistance = distance

    smallestDistance = round(smallestDistance)

    if smallestDistance == 0:
        # xy is directly on a treasure chest!
        chests.remove([x, y])
        return 'You have found a sunken treasure chest!'
    else:
        if smallestDistance < 10:
            board[x][y] = str(smallestDistance)
            return 'Treasure detected at a distance of %s from the sonar device.' %
(smallestDistance)
        else:
            board[x][y] = 'X'
            return 'Sonar did not detect anything. All treasure chests out of range.'

def enterPlayerMove(previousMoves):
    # Let the player enter their move. Return a two-item list of int xy coordinates.
    print('Where do you want to drop the next sonar device? (0-59 0-14) (or type quit)')
    while True:
        move = input()
        if move.lower() == 'quit':
            print('Thanks for playing!')
            sys.exit()

        move = move.split()
        if len(move) == 2 and move[0].isdigit() and move[1].isdigit() and
isOnBoard(int(move[0]), int(move[1])):
            if [int(move[0]), int(move[1])] in previousMoves:
                print('You already moved there.')
                continue
            return [int(move[0]), int(move[1])]

        print('Enter a number from 0 to 59, a space, then a number from 0 to 14.')

def showInstructions():
    print('''Instructions:
You are the captain of the Simon, a treasure-hunting ship. Your current mission
is to use sonar devices to find three sunken treasure chests at the bottom of
the ocean. But you only have cheap sonar that finds distance, not direction.

Enter the coordinates to drop a sonar device. The ocean map will be marked with
how far away the nearest chest is, or an X if it is beyond the sonar device's
range. For example, the C marks are where chests are. The sonar device shows a
3 because the closest chest is 3 spaces away.

1 2 3
012345678901234567890123456789012
```

```
0 ~~~~`~```~`~``~~~``~`~~``~~~``~`~ 0
1 ~`~`~``~~`~```~~~```~~`~`~~~`~~~~ 1
2 `~`C``3`~~~~`C`~~~~`````~~``~~~`` 2
3 ````````~~~`````~~~`~`````~`~``~` 3
4 ~`~~~~`~~`~~`C`~``~~`~~~`~```~``~ 4

0123456789012345678901234567890123
1 2 3
(In the real game, the chests are not visible in the ocean.)

Press enter to continue...''')
    input()

    print('''When you drop a sonar device directly on a chest, you retrieve it and the other
sonar devices update to show how far away the next nearest chest is. The chests
are beyond the range of the sonar device on the left, so it shows an X.

1 2 3
0123456789012345678901234567890123

0 ~~~~`~```~`~``~~~``~`~~``~~~``~`~ 0
1 ~`~`~``~~`~```~~~```~~`~`~~~`~~~~ 1
2 `~`X``7`~~~~`C`~~~~`````~~``~~~`` 2
3 ````````~~~`````~~~`~`````~`~``~` 3
4 ~`~~~~`~~`~~`C`~``~~`~~~`~```~``~ 4

0123456789012345678901234567890123
1 2 3

The treasure chests don't move around. Sonar devices can detect treasure chests
up to a distance of 9 spaces. Try to collect all 3 chests before running out of
sonar devices. Good luck!

Press enter to continue...''')
    input()



print('S O N A R !')
print()
print('Would you like to view the instructions? (yes/no)')
if input().lower().startswith('y'):
    showInstructions()

while True:
    # Game setup
    sonarDevices = 20
    theBoard = getNewBoard()
    theChests = getRandomChests(3)
    drawBoard(theBoard)
    previousMoves = []

    while sonarDevices > 0:
        # Show sonar device and chest statuses.
        print('You have %s sonar device(s) left. %s treasure chest(s) remaining.' %
(sonarDevices, len(theChests)))

        x, y = enterPlayerMove(previousMoves)
        previousMoves.append([x, y]) # We must track all moves so that sonar devices can be
updated.

        moveResult = makeMove(theBoard, theChests, x, y)
```

```
        if moveResult == False:
            continue
        else:
            if moveResult == 'You have found a sunken treasure chest!':
                # Update all the sonar devices currently on the map.
                for x, y in previousMoves:
                    makeMove(theBoard, theChests, x, y)
            drawBoard(theBoard)
            print(moveResult)

        if len(theChests) == 0:
            print('You have found all the sunken treasure chests! Congratulations and good
game!')
            break

        sonarDevices -= 1

    if sonarDevices == 0:
        print('We\'ve run out of sonar devices! Now we have to turn the ship around and head')
        print('for home with treasure chests still out there! Game over.')
        print(' The remaining chests were here:')
        for x, y in theChests:
            print(' %s, %s' % (x, y))

    print('Do you want to play again? (yes or no)')
    if not input().lower().startswith('y'):
        sys.exit()
```