```python
#!/usr/bin/env python

import curses
import copy
import sys
import time
import random


##################
# INITIALIZATION #
##################


def main(argv):
    if not len(argv) == 2:
        usage()
        exit(1)

    width = 0
    height = 0
    try:
        width = int(argv[0])
        height = int(argv[1])
    except:
        usage()
        exit(2)

    board = createBoard(width, height)
    if board == None:
        usage()
        exit(3)

    try:
        curses.wrapper(snake, board)
    except Exception as err:
        print(err)

    exit(0)


def usage():
    print("USAGE: ./snake.py [WIDTH] [HEIGHT]")
    print("where [HEIGHT] and [WIDTH] are integers between 10 and the bounds of your terminal")


def createBoard(width, height):
    if width < 10 or height < 10:
        return None

    board = []
    for i in range(height + 2):
        board.append([])
        for j in range(width + 2):
            if i == 0 or i == height + 1:
                board[i].append("-")
            elif j == 0 or j == width + 1:
                board[i].append("|")
            else:
                board[i].append(" ")

    board[0][0] = "+"
    board[height + 1][0] = "+"
    board[0][width + 1] = "+"
    board[height + 1][width + 1] = "+"

    return board


##################
# GAME FUNCTIONS #
##################


def snake(stdscr, board):
    height = len(board)
    width = len(board[0])

    maxHeight = curses.LINES - 3
    maxWidth = curses.COLS - 1

    if height >= maxHeight or width >= maxWidth:
        raise Exception("Bounds too large, the max width is " + str(maxWidth - 3) + " and the max height is " + 
str(maxHeight - 3))
```

```

    curses.start_color()
    curses.init_pair(1, curses.COLOR_BLACK, curses.COLOR_GREEN)
    curses.init_pair(2, curses.COLOR_BLACK, curses.COLOR_RED)

    curses.curs_set(False)

    stdscr.clear()
    stdscr.refresh()

    direction = "right"

    snake = []
    for i in range(3):
        snake.append([height // 2, (width // 4) - i])

    fruit = [(height // 2), width - (width // 4)]

    drawScreen(board, snake, fruit, stdscr)

    stdscr.addstr(height + 2, 0, "Press any key to start")
    stdscr.refresh()
    stdscr.getch()
    stdscr.addstr(height + 2, 0, "")

    drawScreen(board, snake, fruit, stdscr)

    while True:
        time.sleep(0.1)

        direction = getDirection(direction, stdscr)
        drawScreen(board, snake, fruit, stdscr)
        if updateGame(board, snake, fruit, direction) == False:
            break

        if snake[0][0] == 0 or snake[0][1] == 0 or snake[0][0] == len(board) - 1 or snake[0][1] == len(board[0]) - 1:
            break

    stdscr.nodelay(0)
    stdscr.addstr(height + 2, 0, "GAME OVER - SCORE: " + str(len(snake) - 3), curses.A_BOLD | curses.A_UNDERLINE)
    stdscr.addstr(height + 3, 0, "Press any key to exit")
```

```python
    stdscr.refresh()
    time.sleep(0.25)

    stdscr.getch()


def drawScreen(board, snake, fruit, stdscr):
    stdscr.clear()

    for i in range(len(board)):
        for j in range(len(board[i])):
            stdscr.addstr(i, j, board[i][j])

    for i, pos in enumerate(snake):
        stdscr.addstr(pos[0], pos[1], " ", curses.color_pair(1) | curses.A_DIM)
        if i == 0:
            stdscr.addstr(pos[0], pos[1], "X", curses.color_pair(1) | curses.A_BOLD)

    stdscr.addstr(fruit[0], fruit[1], " ", curses.color_pair(2) | curses.A_BOLD)

    stdscr.addstr(len(board) + 2, 0, "SCORE: " + str(len(snake) - 3))

    stdscr.refresh()


def updateGame(board, snake, fruit, direction):
    oldSnake = copy.deepcopy(snake)

    if direction == "right":
        snake[0][1] += 1
    elif direction == "left":
        snake[0][1] -= 1
    elif direction == "up":
        snake[0][0] -= 1
    elif direction == "down":
        snake[0][0] += 1

    for i in range(1, len(snake)):
        snake[i] = oldSnake[i - 1]

    if snake[0] == fruit:
        snake.append(oldSnake[-1])
```

```python
            fruit[:] = list(newFruit(snake, fruit, board))

    for i in range(1, len(snake)):
        if snake[0] == snake[i]:
            return False

    return True


def getDirection(direction, stdscr):
    stdscr.nodelay(1)
    key = stdscr.getch()

    if key == curses.KEY_RIGHT and not direction == "left":
        return "right"
    elif key == curses.KEY_LEFT and not direction == "right":
        return "left"
    elif key == curses.KEY_UP and not direction == "down":
        return "up"
    elif key == curses.KEY_DOWN and not direction == "up":
        return "down"
    else:
        return direction

def newFruit(snake, fruit, board):
    height = len(board)
    width = len(board[0])
    newFruitY = random.randrange(1, height - 1)
    newFruitX = random.randrange(1, width - 1)

    if [newFruitY, newFruitX] == fruit:
        return newFruit(snake, fruit, board)

    for i in snake:
        if [newFruitY, newFruitX] == i:
            return newFruit(snake, fruit, board)

    return [newFruitY, newFruitX]


###########
# STARTUP #
```

```
###########


if __name__ == "__main__":
    main(sys.argv[1:])
```