“snake”

// const readline = require("readline");

// const chalk = require("chalk");

// const rl = readline.createInterface({

// input: process.stdin,

// output: process.stdout,

// });

// if (process.stdin.isTTY) process.stdin.setRawMode(true);

// const row = (c) => (n) => c.repeat(n);

// const col = (row) => (n) => (row + "\n").repeat(n); //

// const newLine = () => "\033c";

// const rnd = (min) => (max) => Math.round(Math.random() \* (max - min));

// let state = {

// appleRow: 4,

// appleCol: 10,

// snakeRow: 0,

// snakeCol: 3,

// };

// const addApple = (state) => {

// return (board) => {

// let boardArr = board.split("\n").map((line) => line.split(""));

// return boardArr.map((line, row) => {

// return line.map((point, col) => {

// return row == state.appleRow && col == state.appleCol

// ? chalk.red("0")

// : ".";

// });

// });

// };

// };

// const addSnake = (state) => {

// return (board) => {

// return board.map((line, row) => {

// return line.map((point, col) => {

// return row == state.snakeRow && col == state.snakeCol

// ? chalk.green("8")

// : row == state.appleRow && col == state.appleCol

// ? chalk.red("0")

// : ".";

// });

// });

// };

// };

// const randomApple = (state) => {

// state.appleCol = rnd(0)(80);

// state.appleRow = rnd(0)(35);

// return state;

// };

// // const nextApple = (state) => {

// // const check = (state) => {

// // state.snake.forEach(coord => {

// // if(coord[0] == state.app)

// // });

// // }

// // }

// readline.emitKeypressEvents(process.stdin);

// process.stdin.on("keypress", (x, key) => {

// if (key.name == "left") {

// state.snakeCol = state.snakeCol - 1;

// } else if (key.name == "right") {

// state.snakeCol = state.snakeCol + 1;

// } else if (key.name == "up") {

// state.snakeRow = state.snakeRow - 1;

// } else if (key.name == "down") {

// state.snakeRow = state.snakeRow + 1;

// }

// });

// setInterval(() => {

// // COLOR

// const r = rnd(0)(255);

// const g = rnd(0)(255);

// const b = rnd(0)(255);

// // SNAKE TEXT

// let header = newLine();

// header += chalk.rgb(r, g, b)(row(" ")(35) + "SNAKE" + row(" ")(40));

// // Snake

// console.log(header);

// let newAppleState = randomApple(state);

// let board = addApple(newAppleState)(col(row(".")(80))(35));

// board = addSnake(state)(board);

// let boardString = board.map((line) => line.join("")).join("\n");

// console.log(boardString , "ä̶̱͖̺̝͓͉̠͊̄̒͠͠͝l̵̨̹̪̰̙̺̓̇͐̕̚l̶̨̧̠̦̩̮͓̭͐̂͌ä̴͎̦͎́́̐̏̑͊̉̚d̵̳̞̥̼̱̳̣͓̥͋̏͒́̀̈́͗͜͝͝d̵̩͙͉͇͓̠͈͊͌̇̉̈i̷̢̫̝̣͕̟̅̒͛̇̈́͐̃̚n̶͎͖̥̽̉͛̚", "🅰🅻🅻🅰🅳🅳🅸🅽");

// }, 500);

const readline = require('readline')

const chalk = require('chalk')

readline.emitKeypressEvents(process.stdin);

process.stdin.setRawMode(true);

process.stdin.on('keypress', (str, key) => {

if (key.ctrl && key.name === 'c') process.exit();

if (key.name == "up" && gameState.dir != 'D') gameState.dir = 'U'

if (key.name == "down" && gameState.dir != 'U') gameState.dir = 'D'

if (key.name == "left" && gameState.dir != 'R') gameState.dir = 'L'

if (key.name == "right" && gameState.dir != 'L') gameState.dir = 'R';

});

const row = c => n => c.repeat(n)

const col = row => n => (row + '\n').repeat(n)

const newLine = () => '\033c'

const rnd = (min) => (max) => min + Math.round(Math.random() \* (max - min))

const nextApple = (state) => {

const check = (state) => { //[[7, 7], [7, 6], [7, 5]]

state.snake.forEach(coord => {

if (coord[0] == state.appleX && coord[1] == state.appleY) return false;

})

return true;

}

state.appleX = rnd(0)(14)

state.appleY = rnd(0)(19)

while (check(state) == false) {

state.appleX = rnd(0)(14)

state.appleY = rnd(0)(19)

}

return state

}

const addApple = (state) => (board) => {

return board.map((r, i) => {

return i != state.appleX ? r : r.map((v, j) => {

return j == state.appleY ? chalk.red('') : v

})

})

}

let gameState = {

dir: 'R',

snake: [[7, 7], [7, 6], [7, 5]],

appleX: rnd(0)(20),

appleY: rnd(0)(15)

}

const eatApple = (state) => {

if (state.appleX == state.snake[0][0] && state.appleY == state.snake[0][1]) state = nextApple(state);

else state.snake.pop();

return state

}

const addSnake = (state) => (board) => {

return board.map((r, i) => {

return r.map((v, j) => {

let flag = false

state.snake.forEach(coord => {

if (coord[0] == i && coord[1] == j) {

flag = true

}

})

return flag ? chalk.yellow('O') : v

})

})

}

const moveSnake = (state) => {

if (state.dir === 'R')

state.snake.unshift([state.snake[0][0], (20 + state.snake[0][1] + 1) % 20])

if (state.dir === 'L')

state.snake.unshift([state.snake[0][0], (20 + state.snake[0][1] - 1) % 20])

if (state.dir === "U")

state.snake.unshift([(15 + state.snake[0][0] - 1) % 15, state.snake[0][1]])

if (state.dir === "D")

state.snake.unshift([(15 + state.snake[0][0] + 1) % 15, state.snake[0][1]])

console.log(state)

return state;

}

const gameOver = (state) => {

if( state.snake.filter(coord => coord[0] == state.snake[0][0] && coord[1] == state.snake[0][1]).length > 1 ) {

console.log(chalk.bgRed('GOOD GAME DUDE ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !'))

process.exit()

}

}

let board = col(row('.')(20))(15).split('\n').map((x) => x.split(''))

gameState = nextApple(gameState)

board = addApple(gameState)(board)

console.log(board)

setInterval(() => {

let header = newLine()

header += chalk.green(row(' ')(12) + 'SNAKE' + row(' ')(12))

console.log(header)

let board = col(row('.')(20))(15).split('\n').map((x) => x.split(''))

gameState = moveSnake(gameState);

gameState = eatApple(gameState)

board = addApple(gameState)(board)

board = addSnake(gameState)(board)

console.log(board.map((x) => x.join(' ')).join('\n'))

gameOver(gameState)

}, 100)