Contents

1 Basic Test Results

```
********** TEST START **********

  preparing sub.tar
dos2unix: converting file /tmp/bodek._MLOGg/nand2tet/Project09/roigreenberg/presubmission/testdir/stud/sub.tar/README to Uni
compiling...
Game compiled successfully. I hope it's fun!

********** TEST END **********
```

2 README

```
roigreenberg,inbaravni

==============================================================================

Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il

==============================================================================



                Project 9 - High Level Language

                --------------------------------






Submitted Files

---------------

README - This file.

Main.jack - the main.
SnakeGame.jack - The game
Snake.jack - Represent the snake head in the game
Food.jack - Represent the food in the game
List.jack - A linked List class
Node.jack - A node class

Remarks

-------

To start the game, you should choose the game speed, a number between 0 and 9.
Any other choose will be treated as 9(fastest).
The snake is always moving and can turn with pressing on the arrows keys.
Not that the pressing sometimes need to be little long for the game to recognize the press.
The game will be over if the snake will touch the walls or bite itself

To quit from the game press Q.
```

3 Food.jack

```
class Food {
    field int x;
    field int y;
    //field Food food;

    constructor Food new() {
        let x = 100;
        let y = 100;
        do next();
        return this;
    }

    method void dispose() {
        do Memory.deAlloc(this);
        return;
    }

    method void next() {
        do Screen.setColor(false);
        do Screen.drawRectangle(x, y, x + 4, y + 4);
        do nextX();
        do nextY();
        do Screen.setColor(true);
        do Screen.drawRectangle(x, y, x + 4, y + 4);

        return;
    }

    method boolean hit(int hx,int hy) {
        var int i, j;
        let i = 0;
        let j = 0;

        if ((((x - 1) < hx)&((y - 1) < hy))&(((x + 5) > hx)&((y + 5) > hy))) {
            return true;
        }

        return false;
    }

    method void nextX () {
        while (true) {
            let x = Math.multiply(x , 373) - Math.multiply(Math.divide(Math.multiply(x , 373), 491), 491) + 15;
            if ((x < 496)&(x > 15)){
                return;
            }
        }
        return;
    }

    method void nextY () {
        while (true) {
            let y = Math.multiply(y , 151) - Math.multiply(Math.divide(Math.multiply(y , 151), 227), 227) + 15;
            if ((y < 236)&(y > 15)){
                return;
            }
        }
        return;


    }
}
```

4 List.jack

```
class List {

    field Node head;
    field Node tail;

    constructor List new(int givenX, int givenY){

        var int i;
        let i = 20;
        while (i > 0){
            let i = i - 1;
            do newLink(givenX, givenY + i);
        }
        return this;
    }



    method void newLink(int givenX, int givenY){

        var Node newNode;

        if (head = null){
            let newNode = Node.new(givenX, givenY, null, null);
            let tail = newNode;
        } else {
            let newNode = Node.new(givenX, givenY, head, null);
            let head[3] = newNode;
        }

        let head = newNode;
        do Screen.setColor(true);
        do Screen.drawPixel(givenX, givenY);
        return;
    }


    method void removeLink(){


        var Node nodeToBeTail;
        let nodeToBeTail = tail[3];
        let nodeToBeTail[2] = null;
        do tail.dispose();
        let tail = nodeToBeTail;
        do Screen.setColor(false);
        do Screen.drawPixel(nodeToBeTail[0], nodeToBeTail[1]);
        return;
    }


    method boolean isInList(int givenX, int givenY){

        var Node curNode;
        let curNode = head;

        while (~(curNode = null)){

            if ((curNode[0] = givenX) & (curNode[1] = givenY)){
                return true;
            }
            let curNode = curNode[2];
        }
        return false;
    }


    method void dispose(){

        var Node curNode, tmpNode;
        let curNode = head;

        while (~(curNode = null)){
            let tmpNode = curNode[2];
            do curNode.dispose();
            let curNode = tmpNode;
        }
        do Memory.deAlloc(this);
        return;
    }


}
```

5 Main.jack

```
class Main {
    /** Initializes a new game and starts it. */
    function void main() {
        var SnakeGame game;

        let game = SnakeGame.new();
        do game.game();
        do game.dispose();
        return;
    }


}
```

6 Node.jack

```
class Node {

    field int x, y;
    field Node next, prev;
    //static Node head;


    /*constructor Node new(int givenX, int givenY){

        let x = givenX;
        let y = givenY;

        if (head = null){
            let head = this;
            let next = this;
        } else {
            let next = head;
        }

        let prev = null;

        return this;
    }*/


    constructor Node new(int givenX, int givenY, Node givenNext, Node givenPrev){

        let x = givenX;
        let y = givenY;
        let next = givenNext;
        let prev = givenPrev;

        return this;
    }


    method void dispose(){

        do Memory.deAlloc(this);

        return;
    }

}
```

7 Snake.jack

```
class Snake {
    field int x;
    field int y;
    field int curDir;
    field List body;
    field Food food;
    field int count, grow;

    constructor Snake new(int X, int Y) {
        let x = X;
        let y = Y;
        let curDir = 0;
        let count = 0;
        let grow = 0;
        let body = List.new(x,y);
        let food = Food.new();

        //do Screen.setColor(true);
        //do Screen.drawPixel(x, y);
        return this;
    }

    method void dispose() {
        do body.dispose();
        do food.dispose();
        do Memory.deAlloc(this);
        return;
    }


    method boolean move(String dir) {
        if (dir = 0) {
            if (~(curDir = 1)) {
                let y = y - 1;
                let curDir = dir;
            }
            else {
                let y = y + 1;
            }
        }
        if (dir = 1) {
            if (~(curDir = 0)) {
                let y = y + 1;
                let curDir = dir;
            }
            else {
                let y = y - 1;
            }
        }
        if (dir = 3) {
            if (~(curDir = 2)) {
                let x = x + 1;
                let curDir = dir;
            }
            else {
                let x = x - 1;
            }
        }
        if (dir = 2) {
            if (~(curDir = 3)) {
                let x = x - 1;
                let curDir = dir;
            }
            else {
                let x = x + 1;
            }

        }


        if (x = 15) {
            return false;
        }
        if (x = 500) {
            return false;
        }
        if (y = 15) {
            return false;
        }
        if (y = 240) {
            return false;
        }
        if (body.isInList(x,y)){
            return false;
        }
        do body.newLink(x,y);
        if (hit()) {
            let count = count + 1;
            let grow = 10;
            do food.next();
        } else {
            if(grow > 0){
                let grow = grow - 1;
            } else {
                do body.removeLink();
            }

        }
        return true;
    }

    method boolean hit() {

        return food.hit(x,y);
    }

}
```

8 SnakeGame.jack

```
class SnakeGame {

    // The Snake
    field Snake snake;
    //field List body;
    field boolean cont;
    field char key;
    field int fx;
    field int score;




    // The snake's movement direction
    field int direction; // 0=none,1=up,2=down,3=left,4=right

    /** Constructs a new Snake Game. */
    constructor SnakeGame new() {
        let cont = true;
        let direction = 0;
        let score = 0;

        return this;
    }

    /** Deallocates the object's memory. */
    method void dispose() {
        do snake.dispose();
        do Memory.deAlloc(this);
        return;
    }

    method void delay(int speed) {
        var int time1, time2;
        var char tempKey;
        if (speed < 0){
            let speed = 1;
        }
        if (speed > 9) {
            let speed = 10;
        }
        let time1 = 1000/speed;
        let time2 = 1000/speed;

        while (time1 > 0) {
            while (time2 > 0) {
                let tempKey = Keyboard.keyPressed();
                if (~(tempKey = 0)) {
                    let key = tempKey;
                }
                let time2 = time2 - 1;
            }
            let time1 = time1 - 1;
        }
        return;
    }

    method void game() {

        var boolean cont;
        var int i;
        let cont = true;


        let cont = false;
        do printStart();
        let snake = Snake.new(267,127);

        while (key = 0) {
            let key = Keyboard.keyPressed();
        }

        let i = 0;
        while (i < 30) {
            do Output.moveCursor(5,25+i);
            let i = i + 1;
        }


        do run(snake, (key - 47));
        do snake.dispose();

        return;
    }

    method void run(Snake s, int speed) {

        var boolean exit;
        var int direction;
        var boolean cont, hit;
        let direction = 0;
        let exit = false;

        while (~exit) {


            // waits for a key to be pressed.
            while (key = 0) {
                let key = Keyboard.keyPressed();
                do delay(speed);
                if(~(s.move(direction))){
                    do Output.moveCursor(10,25);
                    do Output.printString("Game Over :(");
                    return;
                }
                do Output.moveCursor(0,38);
                do Output.moveCursor(0,37);
                do Output.moveCursor(0,36);
                do Output.printInt(s[5]);

            }

            if (key = 140) {
                let exit = true;
            }
            if (key = 131) {
                let direction = 0;
            }
            if (key = 133) {
                let direction = 1;
            }
            if (key = 130) {
                let direction = 2;
            }
            if (key = 132) {
                let direction = 3;
            }

            // waits for the key to be released.
            while (~(key = 0)) {
                let key = Keyboard.keyPressed();
                do delay(speed);

                if(~(s.move(direction))){
                    do Output.moveCursor(10,25);
                    do Output.printString("Game Over :(");
                    return;
                }
                do Output.moveCursor(0,38);
                do Output.moveCursor(0,37);
                do Output.moveCursor(0,36);
                do Output.printInt(s[5]);
            }
        }

        return;
    }

    method void printStart(){
        do Screen.clearScreen();
        do Screen.setColor(true);
        do Screen.drawLine(15, 15, 15, 240);
        do Screen.drawLine(500, 15, 500, 240);
        do Screen.drawLine(15, 15, 500, 15);
        do Screen.drawLine(15, 240, 500, 240);
        do Output.moveCursor(5,25);
        do Output.printString("Please choose speed (0-9):");
        do Output.moveCursor(0,30);
        do Output.printString("Score:");


        return;
    }



}
```