Course: CS3642 Artificial Intelligence Section W01
Student name: Aaron Cummings
Student ID: 000857610
Assignment #: 2
Due Date: March 21, 2021
Signature: 
Score:

Design of Algorithm:

```
class algorithm{
    double temp;
    double tempMax = value;
    double tempMin = value;
    double delta;
    double eOld;
    double eNew;
    int iMax = value;
    Random rand = new Random();
    NodeObject tempMove = new NodeObject();



    public void annealingAlg(NodeObject Y){
        for (temp = tempMax; temp >= tempMin; temp = nextTemp(temp)){
            for (int i = 0 ; i <= iMax; i++){
                tempMove = Y.successorFunction();
                eOld = Y.energyFunction(Y.state);
                eNew = Y.energyFunction(tempMove);
                delta = eNew - eOld;
                if (delta<0){
                    double random = rand.nextDouble();
                    //System.out.println(random);
                    if (random > Math.exp(delta/temp)){
                        //reject bad move
                    }else{
                        //accept bad move
                        Y.acceptSuccessor(tempMove);
                    }
                }else{
                    //always accept good moves
                    Y.acceptSuccessor(tempMove);

                }
            }
        }

    }

    //reduces temp
    private double nextTemp(double tempIn){
        return tempIn*0.99;
    }
}
```

Tasks my Algorithm is solving:

For part one of the assignment my algorithm is solving an 8 puzzle. The energy function here is based on the number of positions correct. For the second part of the assignment my algorithm is finding a goal word form a randomized collection of the letter that make up that word. The only moves possible are switching adjacent letters in the randomized collection of letters . The energy function is also based on the number of letters in their correct position.

Source Code Part 1:

```
import java.util.*;

//run simulated annealing algorithm to solve 8 puzzle problem
public class main {
    public static void main (String[] args){

        //Configuring console output for # of attempts
        int attempts = 1;
        algorithm alg = new algorithm();
        double[] scores = new double[attempts];
        eightPuzzle[] puzzleArray = new eightPuzzle[attempts];

        for (int i = 0; i<attempts; i++){
            puzzleArray[i] = new eightPuzzle();
        }
        for (int j = 0; j<attempts;j++){
            System.out.println("Starting State in trial "+j);
            puzzleArray[j].toConsole();
            alg.annealingAlg(puzzleArray[j]);
            System.out.println("Ending State in trial "+j);
            puzzleArray[j].toConsole();
            scores[j] = puzzleArray[j].E;
            System.out.println(puzzleArray[j].E);
            System.out.println("\n\n\n\n");
        }
        int
count0=0,count10=0,count20=0,count30=0,count40=0,count50=0,count60=0,count70=
0,count80=0;

        for (int h = 0; h < attempts; h++){
            switch((int) scores[h]){
                case(0):
                    count0++;
                    break;
                case(10):
                    count10++;
                    break;
                case(200):
                    count20++;
                    break;
                case(30):
                    count30++;
                    break;
                case(40):
                    count40++;
                    break;
```

```
                case(50):
                    count50++;
                    break;
                case(60):
                    count60++;
                    break;
                case(70):
                    count70++;
                    break;
                case(80):
                    count80++;
                    break;
                default:
                    break;
            }
        }
        System.out.println("Number of Trials out of "+attempts+" with...");
        System.out.println("0 places correct: "+count0);
        System.out.println("1 places correct: "+count10);
        System.out.println("2 places correct: "+count20);
        System.out.println("3 places correct: "+count30);
        System.out.println("4 places correct: "+count40);
        System.out.println("5 places correct: "+count50);
        System.out.println("6 places correct: "+count60);
        System.out.println("7 places correct: "+count70);
        System.out.println("8 places correct: "+count80);

    }
}

//generates random starting state for an 8 puzzle
//0 represents the 'blank tile'
class eightPuzzle{
    int[][] puzzleArray = new int[3][3];
    int[][] goalArray = new int[3][3];
    int[] indexOfEmptySpace = new int[2];
    double E;
    List<int[][]> moveStates = new ArrayList<int[][]>();

    //true if move is possible
    // 0 up, 1 down, 2 right, 3 left
    boolean[] moves = new boolean[4];

    //Initializes a puzzle with random positions for values 0-8
    eightPuzzle(){
        goalArray[0][0] = 1;
        goalArray[0][1] = 2;
        goalArray[0][2] = 3;
        goalArray[1][0] = 8;
        goalArray[1][1] = 0;
        goalArray[1][2] = 4;
        goalArray[2][0] = 7;
        goalArray[2][1] = 6;
        goalArray[2][2] = 5;

        List<Integer> numPool = new ArrayList<Integer>(9);
        for (int i = 0; i <= 8; i++){
```

```
            numPool.add(i);
        }
        Random rand = new Random();
        int nextIndex;
        for (int j = 0; j <= 2; j++){
            for (int k = 0; k <= 2; k++){
                nextIndex = rand.nextInt(numPool.size());
                puzzleArray[j][k] = numPool.get(nextIndex);
                numPool.remove(nextIndex);
            }
        }
        findEmptyIndex();

        this.identifyMoves();

        E = energyFunction(puzzleArray);

    }

    //Prints the current state to the console
    public void printPuzzle(){
        for (int j = 0; j <= 2; j++) {
            for (int k = 0; k <= 2; k++) {
                System.out.print(puzzleArray[j][k]);
            }
            System.out.println();
        }
    }

    //Returns the index of the black position of the current state
    public void findEmptyIndex(){
        for (int j = 0; j <= 2; j++) {
            for (int k = 0; k <= 2; k++) {
                if (puzzleArray[j][k] == 0){
                    indexOfEmptySpace[0] = j; indexOfEmptySpace[1] = k;
                }
            }
        }
    }

    //returns the energy of the current state
    public double energyFunction(int arrayIn[][]){
        double energy = 0;
        for (int i =0; i<=2;i++) {
            for (int j = 0; j <= 2; j++) {
            if (goalArray[i][j] == arrayIn[i][j] && arrayIn[i][j] != 0){
                energy = energy + 10;
            }
            }
        }
        return energy;
    }

    //updates the list of possible moves for the current state, boolean
values
    public void identifyMoves(){
        switch (indexOfEmptySpace[0]){
```

```
            case (0):
                moves[0] = false;
                moves[1] = true;
                break;
            case (1):
                moves[0] = true;
                moves[1] = true;
                break;
            case (2):
                moves[0] = true;
                moves[1] = false;
                break;
            default:
                break;
        }
        switch (indexOfEmptySpace[1]){
            case (0):
                moves[2] = true;
                moves[3] = false;
                break;
            case (1):
                moves[2] = true;
                moves[3] = true;
                break;
            case (2):
                moves[2] = false;
                moves[3] = true;
                break;
            default:
                break;
        }
    }

    //prints the current state and details of the current state to the
console
    public void toConsole(){
        printPuzzle();
        //System.out.println("The index of 0 is: "+indexOfEmptySpace[0]+",
"+indexOfEmptySpace[1]);
        //System.out.println("Can 0 move...");
        //System.out.println("Up..."+moves[0]);
        //System.out.println("Down..."+moves[1]);
        //System.out.println("Right..."+moves[2]);
        //System.out.println("Left..."+moves[3]);
        System.out.println("energy of this array: "+E);
        System.out.println();
    }

    //returns a random successor state of the current state
    public int[][] successorFunction(){
        int[][] successor = new int[3][3];
        for (int i =0; i<=2; i++){
            for (int j=0; j<=2; j++){
                successor[i][j] = puzzleArray[i][j];
            }
        }
        boolean moveFound = false;
```

```
        Random rand = new Random();
        int y = this.indexOfEmptySpace[0];
        int x = this.indexOfEmptySpace[1];
        int temp;
        while (moveFound == false){
            int move = rand.nextInt(4);
            //System.out.println("move number"+move);
            //0 up,1 down,2 right,3 left
            switch (move){
                case (0):
                    if (moves[0]==true) {

                        temp = successor[y-1][x];
                        successor[y-1][x] = successor[y][x];
                        successor[y][x] = temp;
                        moveFound = true;
                    }

                    break;
                case (1):
                    if (moves[1]==true) {
                        temp = successor[y+1][x];
                        successor[y+1][x] = successor[y][x];
                        successor[y][x] = temp;
                        moveFound = true;
                    }

                    break;
                case (2):
                    if (moves[2]==true) {
                        temp = successor[y][x+1];
                        successor[y][x+1] = successor[y][x];
                        successor[y][x] = temp;
                        moveFound = true;
                    }

                    break;
                case (3):
                    if (moves[3]==true) {
                        temp = successor[y][x-1];
                        successor[y][x-1] = successor[y][x];
                        successor[y][x] = temp;
                        moveFound = true;
                    }
                    break;
                default:
                    break;
            }
        }
        return successor.clone();
    }

    //updates the current state to a successor
    public void acceptSuccesor(int[][] newArray){
        for (int i =0; i<=2; i++){
            for (int j=0; j<=2; j++){
                puzzleArray[i][j] = newArray[i][j];
```

```
            }
        }
        this.findEmptyIndex();
        this.identifyMoves();
        this.E = energyFunction(this.puzzleArray);

    }
}

//Simulated annealing algorithm
class algorithm{
    double temp;
    double tempMax = 10000;
    double tempMin = 1;
    double delta;
    double eOld;
    double eNew;
    int iMax = 100;
    Random rand = new Random();
    int[][] tempMove = new int[3][3];



    public void annealingAlg(eightPuzzle Y){
        for (temp = tempMax; temp >= tempMin; temp = nextTemp(temp)){
            for (int i = 0 ; i <= iMax; i++){
                tempMove = Y.successorFunction().clone();
                eOld = Y.E;
                eNew = Y.energyFunction(tempMove);
                delta = eNew - eOld;
                if (delta<0){
                    double random = rand.nextDouble();
                    if (random > Math.exp(delta/temp)){
                        //reject bad move
                    }else{
                        //accept bad move
                        Y.acceptSuccesor(tempMove.clone());
                    }
                }else{
                    //always accept good moves
                    Y.acceptSuccesor(tempMove.clone());

                }
            }
        }

    }
    private double nextTemp(double tempIn){
        return tempIn*0.99;
    }
}
```

Source Code Part 2:

```
import java.util.*;

//Finds a goal word from a random order of letters, can only switch adjacent
```

```
letters.
public class RealWorldImplementation {
    public static void main(String[] args) {
        String goal = "Hydrostatics";
        System.out.println("Goal word:       "+goal);
        scrambledWord testWord = new scrambledWord(goal);
        System.out.println("Starting state:  "+testWord.toString());
        System.out.println("Starting Energy:
"+testWord.energyFunction(testWord.scrambled));
        wordalgorithm alg = new wordalgorithm();
        alg.annealingAlg(testWord);
        System.out.println("Result:          " +testWord.toString());
        System.out.println("Result Energy:
"+testWord.energyFunction(testWord.scrambled));
    }
}
    //Node class for current state
     class scrambledWord{
        String goalWord;
        Random rand = new Random();
        List<Character> scrambled = new ArrayList<>();
        List<Character> characterList = new ArrayList<>();

        //Generates a random starting position
        public scrambledWord(String word){
            goalWord = word;
            char[] letters = goalWord.toCharArray();
            for (int i = 0 ; i < letters.length; i++){
                characterList.add(letters[i]);
            }

            List<Character> characterListTemp = new ArrayList<>();
            for (int i = 0; i < characterList.size() ; i++){
                characterListTemp.add(characterList.get(i));
            }

            while (characterListTemp.size() != 0 ){
                int j = rand.nextInt(characterListTemp.size());
                scrambled.add(characterListTemp.get(j));
                characterListTemp.remove(j);
            }

        }

        //returns current state as a string
        public String toString(){
            String temp = new String();
            for (int i = 0; i < scrambled.size() ; i++){
                temp = temp + scrambled.get(i);
            }
            return temp;
        }

        //returns the energery of the current state
        public double energyFunction(List<Character> input) {
            double e = 0;
            for (int i = 0 ; i < input.size() ; i++){
```

```
                if (input.get(i) == characterList.get(i) ){
                    e= e +1;
                }
            }
            return e;
        }

        //returns a random successor state for the current state
        public List<Character> successorFunction(){
            List<Character> temp = new ArrayList<>();
            for (int i = 0; i < scrambled.size() ; i++){
                temp.add(scrambled.get(i));
            }
            int secondChar;
            int firstChar = rand.nextInt(scrambled.size()-1);
            if (firstChar == scrambled.size()-1){
                secondChar = firstChar-1;
            } else if (firstChar == 0) {
                secondChar = 1;
            }else{
                if (rand.nextDouble()< .5){
                    secondChar = firstChar - 1;

                }else{
                    secondChar = firstChar + 1;
                }
            }

            Character tempChar;
            tempChar = temp.get(firstChar);
            temp.set(firstChar, temp.get(secondChar));
            temp.set(secondChar, tempChar);
            return temp;
        }

        //set the current state to a successor
        public void acceptSuccessor(List<Character> successor){
            scrambled = successor;
        }

    }

    //Simulated annealing algorithm
     class wordalgorithm{
        double temp;
        double tempMax = 300000;
        double tempMin = .01;
        double delta;
        double eOld;
        double eNew;
        int iMax = 55;
        Random rand = new Random();
        List<Character> tempMove = new ArrayList<>();



        public void annealingAlg(scrambledWord Y){
```

```
            for (temp = tempMax; temp >= tempMin; temp = nextTemp(temp)){
                for (int i = 0 ; i <= iMax; i++){
                    tempMove = Y.successorFunction();
                    eOld = Y.energyFunction(Y.scrambled);
                    eNew = Y.energyFunction(tempMove);
                    delta = eNew - eOld;
                    if (delta<0){
                        double random = rand.nextDouble();
                        //System.out.println(random);
                        if (random > Math.exp(delta/temp)){
                            //reject bad move
                        }else{
                            //accept bad move
                            Y.acceptSuccessor(tempMove);
                        }
                    }else{
                        //always accept good moves
                        Y.acceptSuccessor(tempMove);

                    }
                }
            }

        }
        private double nextTemp(double tempIn){
            return tempIn*0.99;
        }
    }
```