using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace antImageCompressionSystem

{

class Program

{

static void Main(string[] args)

{

string segment = null;

int placeCounter = 0;

var reader = new StreamReader(File.OpenRead(@"C:\Users\Matthew Mouring\Desktop\Matthew\lol\_40crop\_csv.csv"));

List<string> imageList = new List<string>();

while (!reader.EndOfStream)

{

var line = reader.ReadLine();

imageList.Add(line);

}

reader.Close();

string[] imageArray = imageList.ToArray();

int xValue = (imageArray[1].Length + 1) / 2;

int yValue = imageArray.Length;

byte[,] image = new byte[yValue, xValue];

for (int g = 0; g < image.GetLength(0); g++)

{

placeCounter = 0;

for (int f = 0; f < image.GetLength(1); f++)

{

segment = imageArray[g].Substring(placeCounter, 1);

image[g, f] = Byte.Parse(segment);

placeCounter += 2;

}

}

//ANTS PROXIMITY AWARENESS

int[,] proxy = new int[yValue, xValue];

//ANTS CHECKING SMELL DENSITY

int[,] smell = new int[yValue, xValue];

//TRACK THE LEVELS OF THE SMELL

int[,] pheromone = new int[yValue, xValue];

//MIRROR OF THE IMAGE

string[,] mirror = new string[yValue, xValue];

//INITIALIZAE ALL ARRAYS

for (int g = 0; g < image.GetLength(0); g++)

{

for (int f = 0; f < image.GetLength(1); f++)

{

proxy[g, f] = 0;

smell[g, f] = 0;

pheromone[g, f] = 0;

mirror[g, f] = "BU";

}

}

//GET THE POSSIBLE MOVE LOCATIONS

int[,] fourMove = new int[,]{

{0, 0, 0, 0},

{0, 0, 0, 0}

};

//RELATIVE DIRECTIONS

int[,] fiveMove = new int[,]{

{0, 0, 0, 0, 0},

{0, 0, 0, 0, 0}

};

//NUMBER OF ANTS ON AN IMAGE

int numOfAnts = 20;

//THE NUMBER OF TICKS

int numOfRuns = 500000;

//CHECK WHATS AROUND EACH ANT

int[,] sense = new int[,]{

{0, 0, 0, 0, 0, 0, 0, 0}, //X

{0, 0, 0, 0, 0, 0, 0, 0} //Y

};

//FOR CLEARING PAST SENSES

int[,,] oldSense = new int[numOfAnts, 2, 8];

int i = 0, x = 0, y = 0, n = 0;

//CREATE ANTS AND THEIR RECORDS

int[,] ants = new int[numOfAnts, 2];

/\*

\*SET THE RECORDING PROPERLY TO FINISH PROJECT

\*/

List<string> completeAntRecord = new List<string>();

string[] antRecords = new string[numOfAnts];

//INITIALIZE ALL VALUES OF ANTS AND RECORDS TO DEFAULT VALUES

for (i = 0; i < numOfAnts; i++)

{

ants[i, 0] = 2001;

ants[i, 1] = 2001;

antRecords[i] = null;

}

//ANTS CAN MOVE IN FOUR DIRECTIONS UP, RIGHT, LEFT, DOWN

int[] direction = { 0, 1, 2, 3 };

//AFTER THEY FIND FOOD, THEY MOVE IN A RELATIVE DIRECTION, FORWARD, RIGHT, LEFT

int[] relativeDirection = { 0, 0, 0, 0, 0 };

//TRACKS WHAT THE FORWARD VALUE OF EACH ANT COULD BE FOR RELATIVE COMPUTATIONS

int[] forward = new int[numOfAnts];

int[] relativeFront = new int[numOfAnts];

int[] relativeForward = new int[numOfAnts];

//CONTROLS WHAT EACH STAGE OF MOVEMENT DOES

string[] control = new string[numOfAnts];

//KEEP ANTS FROM STARTING IN SAME LOCATIONS

int checkMatches = 0;

Random randomNum = new Random();

int ranNum = 0;

//GIVE RANDOM LOCATIONS TO THE ANTS

for (i = 0; i < numOfAnts; i++)

{

ranNum = randomNum.Next(xValue);

ants[i, 0] = ranNum;

x = ranNum;

ranNum = randomNum.Next(yValue);

ants[i, 1] = ranNum;

y = ranNum;

//MAKE SURE THE ANTS DO NOT STACK ON EACHOTHER

if (i >= 1)

{

for (int k = 0; k <= i; k++)

{

if (x == ants[k, 0] && y == ants[k, 1])

{

checkMatches++;

if (checkMatches >= 2)

{

while (x == ants[k, 0] && y == ants[k, 1])

{

ranNum = randomNum.Next(xValue);

ants[k, 0] = ranNum;

ranNum = randomNum.Next(yValue);

ants[k, 1] = ranNum;

}

}

}

}

checkMatches = 0;

}

x = ants[i, 0];

y = ants[i, 1];

//CHECK FOR THE LOCATION FOR FIRST COORDINATES

getFirstCoord(i, x, y, antRecords, image, mirror, sense, oldSense, proxy, smell, pheromone, control, ants);

}

//BEGIN MOVING THE ANTS

/\*

\* PRIORITY SEARCHES

\* 1. UNEXPLORED

\* 2. LOWER PROXIMITY VALUES

\* 3. HIGHER SMELL VALUES

\* 4. SAME DIRECTION

\* 5. RANDOM

\*/

for (n = 0; n < numOfRuns; n++)

{

//LOWER THE PHEROMONE LEVELS

for (int g = 0; g < pheromone.GetLength(0); g++)

{

for (int f = 0; f < pheromone.GetLength(1); f++)

{

switch (pheromone[g, f])

{

case 1:

pheromone[g, f] = 0;

break;

case 2:

pheromone[g, f] = 1;

break;

default:

break;

}

}

}

//MOVE ANTS

for (i = 0; i < numOfAnts; i++)

{

if (control[i] == "new")

{

newAndFirstMoves(x, y, fourMove, image, smell, proxy, pheromone, mirror, i, ants, direction, forward, randomNum, ranNum);

}

else if (control[i] == "first")

{

newAndFirstMoves(x, y, fourMove, image, smell, proxy, pheromone, mirror, i, ants, direction, forward, randomNum, ranNum);

}

else if (control[i] == "relative")

{

establishRelativeFront(i, forward, relativeDirection, relativeFront);

relativeMovement(x, y, fiveMove, image, smell, proxy, pheromone, mirror, i, ants, relativeDirection, relativeFront, forward, relativeForward, randomNum, ranNum);

}

checkLocation(i, x, y, antRecords, image, mirror, sense, oldSense, proxy, smell, pheromone, control, ants, forward, relativeForward, completeAntRecord);

if (n == numOfRuns - 1)

{

if (antRecords[i] != null)

{

completeAntRecord.Add(antRecords[i]);

antRecords[i] = null;

}

}

}

}

string[] individualAntRecords = completeAntRecord.ToArray();

double numberOfBits = 0;

string moveSegment = "111";

string coordSegment = null;

int enders = 1; //111 - END/START - 7

//WRITE THE STRINGS TO AN EXTERNAL FILE && COUNT THE NUMBER OF BITS IN THE PROGRAM

StreamWriter stringOutput = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\antImageCompressionSystem\CompleteRecords.txt");

StreamWriter stringOutput2 = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\antImageCompressionSystem\movementOutput.txt");

StreamWriter stringOutput3 = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\antImageCompressionSystem\coordinates.txt");

for (int g = 0; g < individualAntRecords.Length; g++)

{

numberOfBits += individualAntRecords[g].Length;

stringOutput.WriteLine(individualAntRecords[g]);

if(individualAntRecords[g].Length - 20 != 0)

{

moveSegment += individualAntRecords[g].Substring(20, (individualAntRecords[g].Length - 20)) + "111";

enders += 1;

}

coordSegment = individualAntRecords[g].Substring(0, 20);

stringOutput3.WriteLine(coordSegment);

}

stringOutput2.WriteLine(moveSegment);

stringOutput.WriteLine();

stringOutput2.WriteLine();

stringOutput3.WriteLine();

stringOutput3.Close();

stringOutput.Close();

stringOutput2.Close();

recreateImage(completeAntRecord, xValue, yValue);

Console.WriteLine("The '111' bits appeared: " + enders);

imageData();

Console.WriteLine("Total Number of bits in Strings: " + numberOfBits);

Console.WriteLine("Total Number of bits in Image: " + (xValue \* yValue));

}

//GET FIRST COORDINATES

static void getFirstCoord(int i, int x, int y, string[] antRecords, byte[,] image, string[,] mirror, int[,] sense, int[,,] oldSense, int[,] proxy, int[,] smell, int[,] pheromone, string[] control, int[,] ants)

{

int y2 = 0;

int x2 = 0;

if (image[y, x] == 0)

{

mirror[y, x] = "BT";

senseTotalRange(x, y, ants, i, sense, mirror);

antRecords[i] = null;

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

control[i] = "new";

}

if (image[y, x] == 1)

{

mirror[y, x] = "WT";

StringBuilder sb = new StringBuilder();

string x\_FORMAT = null;

string y\_FORMAT = null;

x\_FORMAT = Convert.ToString(x, 2).PadLeft(10, '0');

y\_FORMAT = Convert.ToString(y, 2).PadLeft(10, '0');

//sb.Append("(");

sb.Append(x\_FORMAT);

//sb.Append(", ");

sb.Append(y\_FORMAT);

//sb.Append(")");

antRecords[i] += sb.ToString();

senseTotalRange(x, y, ants, i, sense, mirror);

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

addToSmell(sense, smell);

adjustPheromone(sense, pheromone);

control[i] = "first";

}

//FOR TESTING PURPOSES

/\*StringBuilder sb = new StringBuilder();

sb.Append("(");

sb.Append(x + 1);

sb.Append(", ");

sb.Append(y + 1);

sb.Append(")");

antRecords[i] += sb.ToString();\*/

}

//END COORDINATE COLLECT

//CHECK THE NEW LOCATION FOR FOOD

static void checkLocation(int i, int x, int y, string[] antRecords, byte[,] image, string[,] mirror, int[,] sense, int[,,] oldSense, int[,] proxy, int[,] smell, int[,] pheromone, string[] control, int[,] ants, int[] forward, int[] relativeForward, List<string> completeAntRecord)

{

x = ants[i, 0];

y = ants[i, 1];

int y2 = 0;

int x2 = 0;

if (image[y, x] == 0)

{

mirror[y, x] = "BT";

reduceProxy(oldSense, proxy, i);

reduceSmell(oldSense, smell, i);

senseTotalRange(x, y, ants, i, sense, mirror);

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

if (antRecords[i] != null)

{

completeAntRecord.Add(antRecords[i]);

antRecords[i] = null;

}

control[i] = "new";

}

if (image[y, x] == 1)

{

bool explored = false;

//CHECK EXPLORED

while (explored == false)

{

if (mirror[y, x] == "WT" || mirror[y, x] == "BT")

{

control[i] = "new";

if (antRecords[i] != null)

{

completeAntRecord.Add(antRecords[i]);

antRecords[i] = null;

}

explored = true;

}

else

{

/\*Console.WriteLine(x + " , " + y);

Console.WriteLine("------");\*/

mirror[y, x] = "WT";

if (control[i] == "new")

{

if (antRecords[i] != null)

{

completeAntRecord.Add(antRecords[i]);

antRecords[i] = null;

}

StringBuilder sb = new StringBuilder();

string x\_FORMAT = null;

string y\_FORMAT = null;

x\_FORMAT = Convert.ToString(x, 2).PadLeft(10, '0');

y\_FORMAT = Convert.ToString(y, 2).PadLeft(10, '0');

//sb.Append("(");

sb.Append(x\_FORMAT);

//sb.Append(", ");

sb.Append(y\_FORMAT);

//sb.Append(")");

antRecords[i] += sb.ToString();

reduceProxy(oldSense, proxy, i);

reduceSmell(oldSense, smell, i);

senseTotalRange(x, y, ants, i, sense, mirror);

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

addToSmell(sense, smell);

adjustPheromone(sense, pheromone);

control[i] = "first";

explored = true;

break;

}

else if (control[i] == "first")

{

string BINARY\_FORMAT = null;

BINARY\_FORMAT = Convert.ToString(forward[i], 2).PadLeft(3, '0');

StringBuilder sb = new StringBuilder();

//sb.Append("[");

sb.Append(BINARY\_FORMAT);

//sb.Append("] ");

antRecords[i] += sb.ToString();

reduceProxy(oldSense, proxy, i);

reduceSmell(oldSense, smell, i);

senseTotalRange(x, y, ants, i, sense, mirror);

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

addToSmell(sense, smell);

adjustPheromone(sense, pheromone);

control[i] = "relative";

explored = true;

break;

}

else if (control[i] == "relative")

{

string BINARY\_FORMAT = null;

BINARY\_FORMAT = Convert.ToString(relativeForward[i], 2).PadLeft(3, '0');

StringBuilder sb = new StringBuilder();

//sb.Append("<");

sb.Append(BINARY\_FORMAT);

//sb.Append("> ");

antRecords[i] += sb.ToString();

reduceProxy(oldSense, proxy, i);

reduceSmell(oldSense, smell, i);

senseTotalRange(x, y, ants, i, sense, mirror);

for (y2 = 0; y2 < 2; y2++)

{

for (x2 = 0; x2 < 8; x2++)

{

oldSense[i, y2, x2] = sense[y2, x2];

}

}

addToProxy(sense, proxy);

addToSmell(sense, smell);

adjustPheromone(sense, pheromone);

control[i] = "relative";

//RELATIVE PATH

explored = true;

break;

}

}

}

}

}

//END THE CHECK

//MOVE WITH FOUR OPTIONS

static void newAndFirstMoves(int x, int y, int[,] fourMove, byte[,] image, int[,] smell, int[,] proxy, int[,] pheromone, string[,] mirror, int i, int[,] ants, int[] direction, int[] forward, Random randomNum, int ranNum)

{

bool randomNeeded = false;

senseFourMoveRange(i, fourMove, ants, mirror);

for (int f = 0; f < 4; f++)

{

int x2 = fourMove[0, f];

int y2 = fourMove[1, f];

if (x2 != 2001 && y2 != 2001)

{

if (image[y2, x2] == 1 && pheromone[y2, x2] <= 2 && pheromone[y2, x2] != 0 && mirror[y2, x2] == "BU")

{

//MOVE HERE, SET DIRECTIONS

ants[i, 0] = x2;

ants[i, 1] = y2;

if (f == 0)

{

//THEN ANT MOVED UP

forward[i] = 0;

}

else if (f == 1)

{

//THE ANT MOVED RIGHT

forward[i] = 1;

}

else if (f == 2)

{

//THEN ANT MOVE DOWN

forward[i] = 2;

}

else

{

//ANT MOVED LEFT

forward[i] = 3;

}

break;

}

else if (image[y2, x2] == 1 && smell[y2, x2] >= 2 && mirror[y2, x2] == "BU")

{

//MOVE HERE, SET DIRECTIONS

ants[i, 0] = x2;

ants[i, 1] = y2;

if (f == 0)

{

//THEN ANT MOVED UP

forward[i] = 0;

}

else if (f == 1)

{

//THE ANT MOVED RIGHT

forward[i] = 1;

}

else if (f == 2)

{

//THEN ANT MOVE DOWN

forward[i] = 2;

}

else

{

//ANT MOVED LEFT

forward[i] = 3;

}

break;

}

else if (image[y2, x2] == 1 && proxy[y2, x2] < 2 && mirror[y2, x2] == "BU")

{

//MOVE HERE, SET DIRECTIONS

ants[i, 0] = x2;

ants[i, 1] = y2;

if (f == 0)

{

//THEN ANT MOVED UP

forward[i] = 0;

}

else if (f == 1)

{

//THE ANT MOVED RIGHT

forward[i] = 1;

}

else if (f == 2)

{

//THEN ANT MOVE DOWN

forward[i] = 2;

}

else

{

//ANT MOVED LEFT

forward[i] = 3;

}

break;

}

else if (image[y2, x2] == 1 && mirror[y2, x2] == "BU")

{

//MOVE HERE, SET DIRECTIONS

ants[i, 0] = x2;

ants[i, 1] = y2;

if (f == 0)

{

//THEN ANT MOVED UP

forward[i] = 0;

}

else if (f == 1)

{

//THE ANT MOVED RIGHT

forward[i] = 1;

}

else if (f == 2)

{

//THEN ANT MOVE DOWN

forward[i] = 2;

}

else

{

//ANT MOVED LEFT

forward[i] = 3;

}

break;

}

}

if (f == 2)

{

randomNeeded = true;

}

if (randomNeeded == true)

{

//MOVE AT RANDOM

ranNum = randomNum.Next(4);

ants[i, 0] = fourMove[0, ranNum];

ants[i, 1] = fourMove[1, ranNum];

if (fourMove[0, ranNum] == 2001 && fourMove[1, ranNum] == 2001)

{

while (fourMove[0, ranNum] == 2001 && fourMove[1, ranNum] == 2001)

{

ranNum = randomNum.Next(4);

ants[i, 0] = fourMove[0, ranNum];

ants[i, 1] = fourMove[1, ranNum];

x = ants[i, 0];

y = ants[i, 1];

}

}

if (ranNum == 0)

{

//THEN ANT MOVED UP

forward[i] = 0;

}

else if (ranNum == 1)

{

//THE ANT MOVED RIGHT

forward[i] = 1;

}

else if (ranNum == 2)

{

//THEN ANT MOVE DOWN

forward[i] = 2;

}

else

{

//ANT MOVED LEFT

forward[i] = 3;

}

break;

}

}

}

//END FOUR OPTION MOVE

//MOVE WITH RELATIVE DIRECTIONS

static void relativeMovement(int x, int y, int[,] fiveMove, byte[,] image, int[,] smell, int[,] proxy, int[,] pheromone, string[,] mirror, int i, int[,] ants, int[] relativeDirection, int[] relativeFront, int[] forward, int[] relativeForward, Random randomNum, int ranNum)

{

bool matchFound = false;

senseRelativeMoveRange(i, fiveMove, ants, mirror, relativeDirection);

while (matchFound == false)

{

for (int f = 0; f < 5; f++)

{

int x2 = fiveMove[0, f];

int y2 = fiveMove[1, f];

if (x2 != 2001 && y2 != 2001)

{

if (image[y2, x2] == 1 && pheromone[y2, x2] <= 2 && pheromone[y2, x2] != 0 && mirror[y2, x2] == "BU" && relativeFront[i] == relativeDirection[f])

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && smell[y2, x2] >= 2 && mirror[y2, x2] == "BU" && relativeFront[i] == relativeDirection[f])

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && proxy[y2, x2] < 2 && mirror[y2, x2] == "BU" && relativeFront[i] == relativeDirection[f])

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && mirror[y2, x2] == "BU" && relativeFront[i] == relativeDirection[f])

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

matchFound = true;

break;

}

}

}

if (matchFound == true)

{

break;

}

for (int f = 0; f < 5; f++)

{

int x2 = fiveMove[0, f];

int y2 = fiveMove[1, f];

if (x2 != 2001 && y2 != 2001)

{

if (image[y2, x2] == 1 && pheromone[y2, x2] <= 2 && pheromone[y2, x2] != 0 && mirror[y2, x2] == "BU")

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

if (relativeFront[i] == relativeDirection[f])

{

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[0])

{

relativeForward[i] = 2;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[2])

{

relativeForward[i] = 1;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[3])

{

relativeForward[i] = 3;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[4])

{

relativeForward[i] = 4;

forward[i] = relativeDirection[f];

}

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && smell[y2, x2] >= 2 && mirror[y2, x2] == "BU")

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

if (relativeFront[i] == relativeDirection[f])

{

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[0])

{

relativeForward[i] = 2;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[2])

{

relativeForward[i] = 1;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[3])

{

relativeForward[i] = 3;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[4])

{

relativeForward[i] = 4;

forward[i] = relativeDirection[f];

}

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && proxy[y2, x2] < 2 && mirror[y2, x2] == "BU")

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

if (relativeFront[i] == relativeDirection[f])

{

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[0])

{

relativeForward[i] = 2;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[2])

{

relativeForward[i] = 1;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[3])

{

relativeForward[i] = 3;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[4])

{

relativeForward[i] = 4;

forward[i] = relativeDirection[f];

}

matchFound = true;

break;

}

else if (image[y2, x2] == 1 && mirror[y2, x2] == "BU")

{

//MOVE HERE AND ESTABLISH DIRECTION MOVED

ants[i, 0] = x2;

ants[i, 1] = y2;

if (relativeFront[i] == relativeDirection[f])

{

relativeForward[i] = 0;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[0])

{

relativeForward[i] = 2;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[2])

{

relativeForward[i] = 1;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[3])

{

relativeForward[i] = 3;

forward[i] = relativeDirection[f];

}

else if (relativeDirection[f] == relativeDirection[4])

{

relativeForward[i] = 4;

forward[i] = relativeDirection[f];

}

matchFound = true;

break;

}

}

}

if (matchFound == true)

{

break;

}

if (matchFound == false)

{

//MOVE AT RANDOM

ranNum = randomNum.Next(5);

ants[i, 0] = fiveMove[0, ranNum];

ants[i, 1] = fiveMove[1, ranNum];

if (fiveMove[0, ranNum] == 2001 && fiveMove[1, ranNum] == 2001)

{

while (fiveMove[0, ranNum] == 2001 && fiveMove[1, ranNum] == 2001)

{

ranNum = randomNum.Next(5);

ants[i, 0] = fiveMove[0, ranNum];

ants[i, 1] = fiveMove[1, ranNum];

}

}

if (relativeFront[i] == relativeDirection[ranNum])

{

relativeForward[i] = 0;

forward[i] = relativeDirection[ranNum];

}

else if (relativeDirection[ranNum] == relativeDirection[0])

{

relativeForward[i] = 2;

forward[i] = relativeDirection[ranNum];

}

else if (relativeDirection[ranNum] == relativeDirection[2])

{

relativeForward[i] = 1;

forward[i] = relativeDirection[ranNum];

}

else if (relativeDirection[ranNum] == relativeDirection[3])

{

relativeForward[i] = 3;

forward[i] = relativeDirection[ranNum];

}

else if (relativeDirection[ranNum] == relativeDirection[4])

{

relativeForward[i] = 4;

forward[i] = relativeDirection[ranNum];

}

matchFound = true;

}

}

}

//END RELATIVE DIRECTION MOVE

//SENSE ALL SURROUNDINGS(8)

static void senseTotalRange(int x, int y, int[,] ants, int i, int[,] sense, string[,] mirror)

{

x = ants[i, 0];

y = ants[i, 1];

for (int f = 0; f < sense.GetLength(1); f++)

{

switch (f)

{

case 0:

//TO GET COORDINATE FOR UP || POSITION 0

if ((y - 1) >= 0)

{

sense[0, f] = x;

sense[1, f] = y - 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 1:

//TO GET COORDINATE FOR UP-RIGHT || POSITION 1

if ((x + 1) < mirror.GetLength(1) && (y - 1) >= 0)

{

sense[0, f] = x + 1;

sense[1, f] = y - 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 2:

//TO GET COORDINATE FOR RIGHT || POSITION 2

if ((x + 1) < mirror.GetLength(1))

{

sense[0, f] = x + 1;

sense[1, f] = y;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 3:

//TO GET COORDINATE FOR DOWN-RIGHT || POSITION 3

if ((x + 1) < mirror.GetLength(1) && (y + 1) < mirror.GetLength(0))

{

sense[0, f] = x + 1;

sense[1, f] = y + 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 4:

//TO GET COORDINATE FOR DOWN || POSITION 4

if ((y + 1) < mirror.GetLength(0))

{

sense[0, f] = x;

sense[1, f] = y + 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 5:

//TO GET COORDINATE FOR DOWN-LEFT || POSITION 5

if ((x - 1) >= 0 && (y + 1) < mirror.GetLength(0))

{

sense[0, f] = x - 1;

sense[1, f] = y + 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 6:

//TO GET COORDINATE FOR LEFT || POSITION 6

if ((x - 1) >= 0)

{

sense[0, f] = x - 1;

sense[1, f] = y;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

case 7:

//TO GET COORDINATE FOR UP-LEFT || POSITION 7

if ((x - 1) >= 0 && (y - 1) >= 0)

{

sense[0, f] = x - 1;

sense[1, f] = y - 1;

}

else

{

sense[0, f] = 2001;

sense[1, f] = 2001;

}

break;

default:

//ERROR

Console.Write("Search out of scope.");

break;

}

}

}

//END TOTAL SENSE

//GET COORDINATES FOR THE FOUR MOVE OPTIONS(4)

static void senseFourMoveRange(int i, int[,] fourMove, int[,] ants, string[,] mirror)

{

int j = ants[i, 0]; //HANDLE X COORDS

int k = ants[i, 1]; //HANDLE Y COORDS

for (int f = 0; f < 4; f++)

{

switch (f)

{

case 0:

//UP

if ((k - 1) >= 0)

{

fourMove[0, f] = j;

fourMove[1, f] = k - 1;

}

else

{

fourMove[0, f] = 2001;

fourMove[1, f] = 2001;

}

break;

case 1:

//RIGHT

if ((j + 1) < mirror.GetLength(1))

{

fourMove[0, f] = j + 1;

fourMove[1, f] = k;

}

else

{

fourMove[0, f] = 2001;

fourMove[1, f] = 2001;

}

break;

case 2:

//DOWN

if ((k + 1) < mirror.GetLength(0))

{

fourMove[0, f] = j;

fourMove[1, f] = k + 1;

}

else

{

fourMove[0, f] = 2001;

fourMove[1, f] = 2001;

}

break;

case 3:

//LEFT

if ((j - 1) >= 0)

{

fourMove[0, f] = j - 1;

fourMove[1, f] = k;

}

else

{

fourMove[0, f] = 2001;

fourMove[1, f] = 2001;

}

break;

default:

Console.WriteLine("An unexpected error has occured.");

break;

}

}

}

//END COORDINATES COLLECT

//GET COORDINATES FOR THE RELATIVE MOVE OPTIONS (3)

static void senseRelativeMoveRange(int i, int[,] fiveMove, int[,] ants, string[,] mirror, int[] relativeDirection)

{

int j = ants[i, 0];

int k = ants[i, 1];

for (int f = 0; f < 5; f++)

{

switch (relativeDirection[f])

{

case 0:

//UP

if ((k - 1) >= 0)

{

fiveMove[0, f] = j;

fiveMove[1, f] = k - 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 1:

//RIGHT

if ((j + 1) < mirror.GetLength(1))

{

fiveMove[0, f] = j + 1;

fiveMove[1, f] = k;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 2:

//DOWN

if ((k + 1) < mirror.GetLength(0))

{

fiveMove[0, f] = j;

fiveMove[1, f] = k + 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 3:

//LEFT

if ((j - 1) >= 0)

{

fiveMove[0, f] = j - 1;

fiveMove[1, f] = k;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 4:

//UP-RIGHT

if ((j + 1) < mirror.GetLength(1) && (k - 1) >= 0)

{

fiveMove[0, f] = j + 1;

fiveMove[1, f] = k - 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 5:

//DOWN-RIGHT

if ((j + 1) < mirror.GetLength(1) && (k + 1) < mirror.GetLength(0))

{

fiveMove[0, f] = j + 1;

fiveMove[1, f] = k + 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 6:

//DOWN-LEFT

if ((k + 1) < mirror.GetLength(0) && (j - 1) >= 0)

{

fiveMove[0, f] = j - 1;

fiveMove[1, f] = k + 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

case 7:

//UP-LEFT

if ((k - 1) >= 0 && (j - 1) >= 0)

{

fiveMove[0, f] = j - 1;

fiveMove[1, f] = k - 1;

}

else

{

fiveMove[0, f] = 2001;

fiveMove[1, f] = 2001;

}

break;

default:

Console.WriteLine("An unexpected error has occured.");

break;

}

}

}

//END COORDINATES COLLECT

//FIND THE FORWARD MOVEMENT RELATIVE TO ANTS FACE

static void establishRelativeFront(int i, int[] forward, int[] relativeDirection, int[] relativeFront)

{

if (forward[i] == 0)

{

relativeDirection[0] = 3;

relativeDirection[1] = 0;

relativeDirection[2] = 1;

relativeDirection[3] = 7;

relativeDirection[4] = 4;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 1)

{

relativeDirection[0] = 0;

relativeDirection[1] = 1;

relativeDirection[2] = 2;

relativeDirection[3] = 4;

relativeDirection[4] = 5;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 2)

{

relativeDirection[0] = 1;

relativeDirection[1] = 2;

relativeDirection[2] = 3;

relativeDirection[3] = 5;

relativeDirection[4] = 6;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 3)

{

relativeDirection[0] = 2;

relativeDirection[1] = 3;

relativeDirection[2] = 0;

relativeDirection[3] = 6;

relativeDirection[4] = 7;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 4)

{

relativeDirection[0] = 7;

relativeDirection[1] = 4;

relativeDirection[2] = 5;

relativeDirection[3] = 0;

relativeDirection[4] = 1;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 5)

{

relativeDirection[0] = 4;

relativeDirection[1] = 5;

relativeDirection[2] = 6;

relativeDirection[3] = 1;

relativeDirection[4] = 2;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 6)

{

relativeDirection[0] = 5;

relativeDirection[1] = 6;

relativeDirection[2] = 7;

relativeDirection[3] = 2;

relativeDirection[4] = 3;

relativeFront[i] = relativeDirection[1];

}

else if (forward[i] == 7)

{

relativeDirection[0] = 6;

relativeDirection[1] = 7;

relativeDirection[2] = 4;

relativeDirection[3] = 3;

relativeDirection[4] = 0;

relativeFront[i] = relativeDirection[1];

}

else

{

Console.WriteLine("Unexpected Critical Error has occured.");

}

}

//END RELATIVE FORWARD SEARCH

//ADJUST THE PROXIMITY AWARENESS OF THE ANT

static void addToProxy(int[,] sense, int[,] proxy)

{

for (int f = 0; f < sense.GetLength(1); f++)

{

int x2 = sense[0, f];

int y2 = sense[1, f];

if (x2 != 2001 && y2 != 2001)

{

proxy[sense[1, f], sense[0, f]] += 1;

}

else

{

continue;

}

}

}

static void reduceProxy(int[,,] oldSense, int[,] proxy, int i)

{

for (int f = 0; f < oldSense.GetLength(2); f++)

{

int x2 = oldSense[i, 0, f];

int y2 = oldSense[i, 1, f];

if (x2 != 2001 && y2 != 2001)

{

int temp = proxy[y2, x2];

if (temp > 0)

{

proxy[y2, x2] -= 1;

}

else

{

continue;

}

}

else

{

continue;

}

}

}

//END THE AWARENESS SETTER

//ADJUST THE SMELL DENSITY

static void addToSmell(int[,] sense, int[,] smell)

{

for (int f = 0; f < sense.GetLength(1); f++)

{

int x2 = sense[0, f];

int y2 = sense[1, f];

if (x2 != 2001 && y2 != 2001)

{

smell[sense[1, f], sense[0, f]] += 1;

}

else

{

continue;

}

}

}

static void reduceSmell(int[,,] oldSense, int[,] smell, int i)

{

for (int f = 0; f < oldSense.GetLength(2); f++)

{

int x2 = oldSense[i, 0, f];

int y2 = oldSense[i, 1, f];

if (x2 != 2001 && y2 != 2001)

{

int temp = smell[y2, x2];

if (temp > 0)

{

smell[y2, x2] -= 1;

}

else

{

continue;

}

}

else

{

continue;

}

}

}

//END ADJUST SMELL

//ADJUST SMELL PHEROMONE

static void adjustPheromone(int[,] sense, int[,] pheromone)

{

for (int f = 0; f < sense.GetLength(1); f++)

{

int x2 = sense[0, f];

int y2 = sense[1, f];

if (x2 != 2001 && y2 != 2001)

{

pheromone[sense[1, f], sense[0, f]] = 2;

}

else

{

continue;

}

}

}

//END ADJUST PHEROMONE

//BEGIN TO RECREATE THE IMAGE

static void recreateImage(List<string> completedAntRecord, int xValue, int yValue)

{

//COUNTERS FOR THE NUMBER OF EACH STRING TYPE

int fronts = 0; //000 - UP/F - 0

int rights = 0; //001 - R/R - 1

int lefts = 0; //010 - D/L - 2

int fLefts = 0; //011 - L/FL - 3

int fRights = 0; //100 - FR - 4

int[,] newImage = new int[yValue, xValue];

for (int g = 0; g < newImage.GetLength(0); g++)

{

for (int f = 0; f < newImage.GetLength(1); f++)

{

newImage[g, f] = 0;

}

}

string[] individualAntRecords = completedAntRecord.ToArray();

string segmenter = null;

int counter = 0;

int x = 0, y = 0;

int lastDirection = 0;

int[,] fiveOption = new int[,]

{

{0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0 }

};

int[] relativeOptions = { 0, 0, 0, 0, 0 };

for (int g = 0; g < individualAntRecords.Length; g++)

{

//Console.WriteLine(individualAntRecords[g]);

counter = 0;

while (counter < individualAntRecords[g].Length)

{

if (counter == 0)

{

segmenter = individualAntRecords[g].Substring(counter, 10);

x = Convert.ToInt32(segmenter, 2);

counter += 10;

}

if (counter == 10)

{

segmenter = individualAntRecords[g].Substring(counter, 10);

y = Convert.ToInt32(segmenter, 2);

newImage[y, x] = 1;

counter += 10;

}

if (counter >= individualAntRecords[g].Length)

{

break;

}

//GET FIRST DIRECTION

if (counter == 20)

{

segmenter = individualAntRecords[g].Substring(counter, 3);

if (segmenter == "000")

{

//UP

y = y - 1;

newImage[y, x] = 1;

lastDirection = 0;

fronts += 1;

}

else if (segmenter == "001")

{

//RIGHT

x = x + 1;

newImage[y, x] = 1;

lastDirection = 1;

rights += 1;

}

else if (segmenter == "010")

{

//DOWN

y = y + 1;

newImage[y, x] = 1;

lastDirection = 2;

lefts += 1;

}

else if (segmenter == "011")

{

//LEFT

x = x - 1;

newImage[y, x] = 1;

lastDirection = 3;

fLefts += 1;

}

counter += 3;

}

if (counter >= individualAntRecords[g].Length)

{

break;

}

if (counter >= 23)

{

while (counter != individualAntRecords[g].Length)

{

if (counter == individualAntRecords[g].Length)

{

break;

}

//RELATIVE DIRECITONS

if (lastDirection == 0)

{

relativeOptions[0] = 3;

relativeOptions[1] = 0;

relativeOptions[2] = 1;

relativeOptions[3] = 7;

relativeOptions[4] = 4;

}

else if (lastDirection == 1)

{

relativeOptions[0] = 0;

relativeOptions[1] = 1;

relativeOptions[2] = 2;

relativeOptions[3] = 4;

relativeOptions[4] = 5;

}

else if (lastDirection == 2)

{

relativeOptions[0] = 1;

relativeOptions[1] = 2;

relativeOptions[2] = 3;

relativeOptions[3] = 5;

relativeOptions[4] = 6;

}

else if (lastDirection == 3)

{

relativeOptions[0] = 2;

relativeOptions[1] = 3;

relativeOptions[2] = 0;

relativeOptions[3] = 6;

relativeOptions[4] = 7;

}

else if (lastDirection == 4)

{

relativeOptions[0] = 7;

relativeOptions[1] = 4;

relativeOptions[2] = 5;

relativeOptions[3] = 0;

relativeOptions[4] = 1;

}

else if (lastDirection == 5)

{

relativeOptions[0] = 4;

relativeOptions[1] = 5;

relativeOptions[2] = 6;

relativeOptions[3] = 1;

relativeOptions[4] = 2;

}

else if (lastDirection == 6)

{

relativeOptions[0] = 5;

relativeOptions[1] = 6;

relativeOptions[2] = 7;

relativeOptions[3] = 2;

relativeOptions[4] = 3;

}

else if (lastDirection == 7)

{

relativeOptions[0] = 6;

relativeOptions[1] = 7;

relativeOptions[2] = 4;

relativeOptions[3] = 3;

relativeOptions[4] = 0;

}

else

{

Console.WriteLine("Unexpected Critical Error has occured.");

}

for (int f = 0; f < 5; f++)

{

switch (relativeOptions[f])

{

case 0:

//UP

fiveOption[0, f] = x;

fiveOption[1, f] = y - 1;

break;

case 1:

//RIGHT

fiveOption[0, f] = x + 1;

fiveOption[1, f] = y;

break;

case 2:

//DOWN

fiveOption[0, f] = x;

fiveOption[1, f] = y + 1;

break;

case 3:

//LEFT

fiveOption[0, f] = x - 1;

fiveOption[1, f] = y;

break;

case 4:

//UP-RIGHT

fiveOption[0, f] = x + 1;

fiveOption[1, f] = y - 1;

break;

case 5:

//DOWN-RIGHT

fiveOption[0, f] = x + 1;

fiveOption[1, f] = y + 1;

break;

case 6:

//DOWN-LEFT

fiveOption[0, f] = x - 1;

fiveOption[1, f] = y + 1;

break;

case 7:

//UP-LEFT

fiveOption[0, f] = x - 1;

fiveOption[1, f] = y - 1;

break;

default:

Console.WriteLine("An unexpected error has occured.");

break;

}

}

if (counter >= individualAntRecords[g].Length)

{

break;

}

segmenter = individualAntRecords[g].Substring(counter, 3);

if (segmenter == "000")

{

//Console.WriteLine(y + " , " + x);

x = fiveOption[0, 1];

y = fiveOption[1, 1];

newImage[y, x] = 1;

lastDirection = relativeOptions[1];

fronts += 1;

}

else if (segmenter == "001")

{

//Console.WriteLine(y + " , " + x);

x = fiveOption[0, 2];

y = fiveOption[1, 2];

newImage[y, x] = 1;

lastDirection = relativeOptions[2];

rights += 1;

}

else if (segmenter == "010")

{

//Console.WriteLine(y + " , " + x);

x = fiveOption[0, 0];

y = fiveOption[1, 0];

newImage[y, x] = 1;

lastDirection = relativeOptions[0];

lefts += 1;

}

else if (segmenter == "011")

{

//Console.WriteLine(y + " , " + x);

x = fiveOption[0, 3];

y = fiveOption[1, 3];

newImage[y, x] = 1;

lastDirection = relativeOptions[3];

fLefts += 1;

}

else if (segmenter == "100")

{

//Console.WriteLine(y + " , " + x);

x = fiveOption[0, 4];

y = fiveOption[1, 4];

newImage[y, x] = 1;

lastDirection = relativeOptions[4];

fRights += 1;

}

counter += 3;

if (counter >= individualAntRecords[g].Length)

{

break;

}

}

}

}

counter = 0;

}

using (StreamWriter outFile = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\Matthew\testOutput.csv"))

{

for (int g = 0; g < newImage.GetLength(0); g++)

{

string content = "";

for (int f = 0; f < newImage.GetLength(1); f++)

{

content += newImage[g, f].ToString("0") + ",";

}

outFile.WriteLine(content);

}

outFile.Close();

}

Console.WriteLine();

Console.WriteLine("//MOVEMENT");

Console.WriteLine("The '000' bits appeared: " + fronts);

Console.WriteLine("The '001' bits appeared: " + rights);

Console.WriteLine("The '010' bits appeared: " + lefts);

Console.WriteLine("The '011' bits appeared: " + fLefts);

Console.WriteLine("The '100' bits appeared: " + fRights);

}

//END RECREATING IMAGE

//BEGIN TO GATHER VARIOUS IMAGE DATA

static void imageData()

{

var reader = new StreamReader(File.OpenRead(@"C:\Users\Matthew Mouring\Desktop\antImageCompressionSystem\movementOutput.txt"));

int moveBits = 0;

int coordBits = 0;

List<string> movementData = new List<string>();

List<string> coordinatesData = new List<string>();

while (!reader.EndOfStream)

{

var line = reader.ReadLine();

movementData.Add(line);

}

reader.Close();

var reader2 = new StreamReader(File.OpenRead(@"C:\Users\Matthew Mouring\Desktop\antImageCompressionSystem\coordinates.txt"));

while (!reader2.EndOfStream)

{

var line = reader2.ReadLine();

coordinatesData.Add(line);

}

reader2.Close();

//TAKE WHAT WAS READ AND CHANGE LIST TO ARRAYS FOR BIT COUNT

string[] moveData = movementData.ToArray();

string[] coordData = coordinatesData.ToArray();

for (int g = 0; g < moveData.Length; g++)

{

moveBits += moveData[g].Length;

}

for (int g = 0; g < coordData.Length; g++)

{

coordBits += coordData[g].Length;

}

//SEGMENT THE STRINGS FROM THE MOVEMENT OUTPUT FILE FOR DECIMAL CONVERSION

string segMove = null;

int placeInString = 0;

int decVal = 0;

using (StreamWriter outFile = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\Test\_Results\IMG\_40\Run5\IMG40\_movementDecimalValues\_R5.csv"))

{

string decValues = "";

while (placeInString != moveData[0].Length)

{

segMove = moveData[0].Substring(placeInString, 3);

decVal = Convert.ToInt32(segMove, 2);

decValues = decVal.ToString() + ",";

outFile.Write(decValues);

placeInString += 3;

if (placeInString == moveData[0].Length)

{

break;

}

}

outFile.Close();

}

//SEGMENT THE STRINGS FROM THE COORDINATES FILE FOR DECIMAL CONVERSION AND OCCURANCE COUNTING

int[] dirOccurance = new int[4] { 0, 0, 0, 0 };

using (StreamWriter outFile = new StreamWriter(@"C:\Users\Matthew Mouring\Desktop\Test\_Results\IMG\_40\Run5\IMG40\_coordinatesDecimalValues\_R5.csv"))

{

for (int g = 0; g < coordData.Length; g++)

{

string decValues = "";

placeInString = 0;

while (placeInString != coordData[g].Length)

{

segMove = coordData[g].Substring(placeInString, 2);

if (segMove == "00")

{

dirOccurance[0] += 1;

}

else if (segMove == "01")

{

dirOccurance[1] += 1;

}

else if (segMove == "10")

{

dirOccurance[2] += 1;

}

else if (segMove == "11")

{

dirOccurance[3] += 1;

}

decVal = Convert.ToInt32(segMove, 2);

decValues = decVal.ToString() + ",";

outFile.Write(decValues);

placeInString += 2;

if (placeInString == coordData[g].Length)

{

break;

}

}

}

outFile.Close();

}

Console.WriteLine();

Console.WriteLine("//COORDINATES");

Console.WriteLine("The '00' bit appeared: " + dirOccurance[0]);

Console.WriteLine("The '01' bit appeared: " + dirOccurance[1]);

Console.WriteLine("The '10' bit appeared: " + dirOccurance[2]);

Console.WriteLine("The '11' bit appeared: " + dirOccurance[3]);

Console.WriteLine();

Console.WriteLine("//OVERALL STATS");

Console.WriteLine("The Number of Bits in the movements file is: " + moveBits);

Console.WriteLine("The Number of Bits in the coordinates file is: " + coordBits);

}

//END COLLECTION OF DATA

}

}