**CSC 381Image Processing (Java)**

**Project: Median 3X3 Filter**

**Student: Jeffrey Pulinat**

**Due date: soft copy:**

**Hardcopy :**

Algorithm steps:

step 0: - open the input file and output file

- read the image header, the four numbers

- dynamically allocate mirrorFramedAry

- dynamically alloicate tempAry

step 1: read the input file and load onto mirrowframeAry begin at [1,1]

step 2: mirrowFramed the mirrorFramedAr

step 3: process the MirrorframedAry, from left to right and top to bottom

begin at (1, 1)

- neighborAry <- loadNeighbors // get mirrorframedAry[i,j]'s 3 X 3 neighborhoods

- tempAry[i,j] <-- sort neightborAry using selection sort algorithm,

stop after the fifth smallest item is found,

then return the fifth item

- keep tracking the newMin and newMax of tempAry

step 4: repeat step 3 until all pixels are processed

step 5: output the image header (numRows, numCols, newMin, newMax) to outfile

step 6: output the tempAry, begin at [1,1], to outfile

step 7: close input and output files

**B) Source Code**

#include <iostream>

#include <fstream>

#include <string>

#include <stdlib.h>

using namespace std;

static int numRows;

static int numCols;

static int minVal;

static int maxVal;

static int newMin;

static int newMax;

static int neighborAry [9];

static int\*\* mirrorFramedAry;

static int\*\* tempAry;

void mirrorFramed(ifstream& input){

mirrorFramedAry = new int \*[numRows+2];

for(int i=0; i<numRows+2; ++i){

mirrorFramedAry[i] = new int[numCols+2];

}

for(int row = 1; row<= numRows; row++){

for(int col = 1; col<= numCols; col++){

int value;

input >> value;

mirrorFramedAry[row][col] = value;

if(row == 1) mirrorFramedAry[0][col] = value; //mirrors top row

if(row == numRows) mirrorFramedAry[row+1][col] = value; //mirrors bottom row

}

}

for(int side=0; side<=numRows+1; side++){ //mirroring sides

mirrorFramedAry[side][0] = mirrorFramedAry[side][1]; //left side

mirrorFramedAry[side][numCols+1] = mirrorFramedAry[side][numCols]; //right side

}

}

void loadImage(ifstream& input){

int param;

input >> param;

numRows = param;

input >> param;

numCols = param;

input >> param;

minVal = param;

input >> param;

maxVal = param;

mirrorFramed(input);

}

void MEDIAN3X3(int row, int col, int count) {

int i, j, min\_idx;

// Selection sort

//stop after the fifth smallest item is found

for (i = 0; i <=4; i++)

{

// find min index

min\_idx = i;

for (j = i+1; j < 9; j++){

if (neighborAry[j] < neighborAry[min\_idx])

min\_idx = j;

}

// Swap the found minimum element with the first element

int temp = neighborAry[min\_idx];

neighborAry[min\_idx] = neighborAry[i];

neighborAry[i] = temp;

}

int median = neighborAry[4];

if(count==1){ //first iteration

newMin = median;

newMax = median;

} else{ //compare and change min and max

if(newMin>median)newMin = median;

if(newMax<median)newMax = median;

}

tempAry[row][col] = median;

}

void loadNeighbors() { //load 3x3 neighbors

tempAry = new int \*[numRows+2];

for(int i=0; i<numRows+2; ++i){

tempAry[i]= new int[numCols+2];

}

int count =1;

for(int row = 1; row<=numRows; row++){

for(int col = 1; col<=numCols; col++){

neighborAry[0]=mirrorFramedAry[row-1][col-1];

neighborAry[1]=mirrorFramedAry[row-1][col];

neighborAry[2]=mirrorFramedAry[row-1][col+1];

neighborAry[3]=mirrorFramedAry[row][col-1];

neighborAry[4]=mirrorFramedAry[row][col];

neighborAry[5]=mirrorFramedAry[row][col+1];

neighborAry[6]=mirrorFramedAry[row+1][col-1];

neighborAry[7]=mirrorFramedAry[row+1][col];

neighborAry[8]=mirrorFramedAry[row+1][col+1];

MEDIAN3X3(row,col, count);

count++;

}

}

}

int main(int argc, char \*argv[]) {

ifstream input(argv[1]);

string arg1 =argv[1];

loadImage(input);

loadNeighbors();

//making output file

ofstream output(argv[2]);

//new params

output<<numRows<<" "<<numCols<<" "<<newMin<<" "<<newMax<<endl;

for(int row = 1; row<=numRows; row++){ //print out from [1][1] w/o border

for(int col = 1; col<=numCols; col++){

int value = tempAry[row][col];

output<<value<<" ";

}

output<<endl;

}

for (int i = 0; i < numRows; ++i )delete [] tempAry[i];

delete [] tempAry;

for (int i = 0; i < numRows; ++i )delete [] mirrorFramedAry[i];

delete [] mirrorFramedAry;

input.close();

output.close();

}

**OUTPUT**

**Median3X3Out1\_PP.txt (Pretty Print output of median ouput)**

31 40 0 9

1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 2 1 1

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 1 1

1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 1 1

1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 2 1 1 1

1 1 2 1 2 2 2 2 2 2 3 2 3 3 3 3 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 2 2 1 2 2 2 2 1

1 1 2 2 2 2 3 3 3 5 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 5 2 2 2 2 2 2 1 1

1 2 2 2 2 2 3 4 5 6 7 7 7 7 7 7 7 7 7 7 8 7 7 7 7 7 7 7 7 7 7 6 5 2 2 2 2 2 1 1

1 2 2 2 2 2 2 4 5 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 7 5 2 2 2 1 2 1 1

1 2 2 2 2 1 2 5 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 8 7 5 2 1 2 1 2 1 1

1 1 2 2 2 2 1 2 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 5 2 1 2 1 1 1

1 2 2 2 2 1 2 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 2 2 1 2 1 1 1

1 2 3 3 2 2 1 2 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 2 2 1 2 1 1 1

1 1 2 2 2 1 2 5 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 2 1 1 2 1 1

1 2 2 2 2 4 6 9 9 9 9 9 9 9 9 1 1 9 9 9 9 9 9 9 9 9 9 9 8 7 2 1 1 2 1 1

1 1 2 2 2 4 5 6 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 2 1 1 2 1 1

1 1 2 2 4 4 5 6 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 2 1 1 2 1 1

1 2 3 3 3 4 5 6 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 7 1 1 1 2 1 1

1 1 2 2 3 3 5 5 8 9 9 9 9 9 9 9 9 9 9 9 9 8 8 8 9 9 9 9 9 8 7 1 1 1 2 2 1

1 2 2 3 3 3 5 5 8 8 8 8 8 8 8 8 8 8 9 8 9 8 7 7 8 8 8 8 8 7 7 1 1 1 1 2 2 1 1

1 2 1 2 3 3 4 5 6 8 8 7 7 7 7 8 8 8 8 7 8 7 7 7 7 7 7 7 7 7 6 2 1 1 1 2 2 1 1

1 1 2 1 2 2 3 4 3 5 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 5 5 5 5 6 5 5 2 1 2 1 2 1 1 1

1 1 2 1 2 2 2 2 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 2 2 2 2 1 2 1 1 1

1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1

1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 2 1 1 1

1 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 1 1

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 1 1

1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 1 1

1 1 2 2 2 2 2 2 3 2 2 2 2 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1

**Median3X3Out1\_7\_PP.txt (Pretty Print output of Threshold output of median output)**

31 40 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**Median3X3Out2\_PP.txt (Pretty Print of Median output)**

31 40 0 9

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1

2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 1 1 2 2

1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 8 8 8 2 2 1 2 1 2 2 2 2 1 1 2 3 2 1 1 2 2

1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 3 7 8 8 8 7 2 2 2 1 2 2 2 2 1 2 2 3 1 1 1 2 2

1 1 1 1 2 2 1 2 1 1 1 1 1 1 2 7 8 8 8 8 7 7 2 1 1 1 2 2 2 1 1 1 2 1 1 1 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 7 8 8 8 7 7 7 7 7 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 2 1 1 1 7 8 8 8 7 7 7 7 8 8 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1

2 2 1 2 1 2 1 2 1 2 1 2 2 2 7 8 8 8 7 7 8 8 8 8 8 8 8 2 1 1 1 1 1 1 1 1 1 1 1 1

1 2 2 2 1 1 1 2 1 2 1 2 2 7 8 8 8 7 7 9 9 8 8 8 9 9 9 7 1 1 1 1 2 2 1 1 1 1 1

1 2 2 2 1 1 1 1 2 7 8 8 8 7 7 9 9 8 8 7 7 9 8 8 8 8 2 1 1 1 1 1 1 1 1 1

2 2 1 1 1 1 7 8 8 8 8 8 8 8 8 8 8 7 6 9 8 8 8 8 7 2 1 2 1 1 1

2 1 1 1 1 1 1 8 8 8 8 8 8 8 8 8 8 8 8 7 6 8 8 8 8 8 7 7 3 2 1 1 1 1 1

2 2 1 1 1 1 1 1 7 8 8 8 8 8 8 8 8 8 8 7 7 7 8 8 8 8 8 8 8 8 7 3 2 2 1 1 1 1

1 2 2 1 1 1 1 2 8 8 8 8 8 8 8 8 8 8 8 8 7 7 7 8 8 8 8 8 8 7 8 8 8 3 2 1 1 1 1

1 1 1 1 2 8 8 8 8 8 8 8 8 8 9 8 8 7 7 8 8 8 8 8 8 8 8 8 8 8 3 2 1 1 1 1

1 1 8 8 8 8 8 8 8 7 7 9 9 8 8 8 9 9 8 7 8 8 8 8 8 8 8 2 2 1 1 1

1 1 8 8 8 8 8 7 7 8 9 8 8 8 9 9 8 7 6 6 7 8 8 8 7 2 1 1 2 2

2 1 1 1 1 8 8 8 8 8 7 7 8 8 8 8 8 7 8 8 6 7 7 8 7 7 2 2 1

1 2 1 1 1 1 7 8 8 8 8 8 8 8 8 9 9 7 7 8 8 8 8 8 7 2 2 2 2

1 2 1 1 1 1 1 1 8 8 8 8 8 8 8 8 8 8 7 8 9 8 8 8 7 2 1 1 2 2 1 1 1 1 1

2 3 1 1 1 1 1 1 1 1 7 8 8 8 8 8 8 8 8 8 9 9 8 8 7 2 1 1 1 1 2 1 1 1 1 1 1

2 2 1 1 1 1 1 1 1 8 8 7 7 9 9 8 8 8 9 9 8 8 2 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 8 8 7 7 9 9 8 8 8 8 7 2 1 1 1 1 1 1 2 2 1 1 1 1 1

1 2 2 2 1 1 1 1 1 1 1 8 8 7 7 9 9 8 8 7 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1

1 2 2 2 1 2 2 1 1 1 1 1 1 1 8 7 7 9 9 8 7 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1

2 3 2 2 2 2 2 1 1 1 1 7 8 9 9 8 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

2 1 1 1 2 1 1 1 2 1 1 1 1 8 8 8 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 2 2 1 1 1 1 1 1 1 1 1 1

**Median3X3Out2\_8\_PP.txt (Pretty Print output of Threshold output of median output)**

31 40 0 1

1 1 1

1 1 1

1 1 1 1

1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1

1 1 1 1

1 1 1