Name: Alif Rahi
Section: CSCI 381 - Computer Vision / Tues-Thurs 1:40-2:55pm
Project: 4
Due: March 19, 2023

Main algorithm steps:

```
********************************
IV. main(...)
********************************
step 0:  inFile ← open the input file from argv [1]
         Connectness ← argv [2]
         option ← argv [3]
         RFprettyPrintFile, labelFile, propertyFile, deBugFile ← open from argv []
          numRows, numCols, minVal, maxVal ← read from inFile
         zeroFramedAry ← dynamically allocate.
         newLabel ← 0
step 1:  zero2D (zeroFramedAry)
step 2: loadImage (inFile, zeroFramedAry)
step 3: if option == 'y' or 'Y'
         conversion (zeroFramedAry)
step 4: if connectness == 4
           connected4 (zeroFramedAry, newLabel, EQAry, RFprettyPrintFile, deBugFile)
step 5: if connectness == 8
          connected4 (zeroFramedAry, newLabel, EQAry, RFprettyPrintFile, deBugFile)
step 6: labelFile ← output numRows, numCols, newMin, newMax to labelFile
step 7: printImg (zeroFramedAry, labelFile) // Output the result of pass3 inside of zeroFramedAry
```

Source Code

```
#include <iostream>

#include <string>

#include <algorithm>

#include <fstream>

using namespace std;



struct Property

{

   int label, numPixels, minR, maxR, minC;

};



class ccLabel
```

```
{

public:

   int rows, cols, minVal, maxVal, newLabel, trueNumCC, newMin, newMax, thrVal;

   int **zeroFramedArray, *NonZeroNeighborArray, *EQArray;

   Property *CCproperty;


   ccLabel(int r, int c, int mnV, int mxV)

   {

       rows = r;

       cols = c;

       minVal = mnV;

       maxVal = mxV;

       newLabel = 0;

       newMin = 0;

       newMax = trueNumCC;


       zeroFramedArray = new int *[r + 2];


       for (int i = 0; i < r + 2; i++)

       {

           zeroFramedArray[i] = new int[c + 2];

       }


       NonZeroNeighborArray = new int[5];
```

```
    EQArray = new int[r * c / 4];

    int k = 0;

    while (EQArray[k] != '\0')

    {

        EQArray[k] = k;

    }

}



void zero2D(int **arr, int rowz, int colz)

{

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

    {

        for (int j = 0; j < colz; j++)

        {

            arr[i][j] = 0;

        }

    }

}



void loadImage(ifstream &inFile, int **arr)

{

    for (int i = 1; i < rows + 1; i++)

    {

        for (int j = 1; j < cols + 1; j++)
        {
            int val;
            inFile >> val;
            zeroFramedArray[i][j] = val;
        }
    }
}

void conversion()
{
    for (int i = 1; i < rows + 1; i++)
    {
        for (int j = 1; j < cols + 1; j++)
        {
            if (zeroFramedArray[i][j] == 1)
            {
                zeroFramedArray[i][j] = 0;
            }
            else
            {
                zeroFramedArray[i][j] = 1;
            }
        }
    }
```

```
}



int findVal(int arr[], int n)

{

    int res = 1;



    // Pick all elements one by one

    for (int i = 1; i < n; i++)

    {

        int j = 0;

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

            if (arr[i] == arr[j])

                break;



        // If not printed earlier, then print it

        if (i == j)

            res++;

    }

    return res;

}



int findMin(int *arr, int n)

{

    int minV = arr[0];
    for (int i = 1; i < n; i++)
    {
        if (arr[i] < minV)
        {
            minV = arr[i];
        }
    }
    return minV;
}

void connected8pass1()
{
    for (int i = 1; i < rows + 1; i++)
    {
        for (int j = 1; j < cols + 1; j++)
        {
            if (zeroFramedArray[i][j] > 0)
            {
                int a = zeroFramedArray[i - 1][j - 1];
                int b = zeroFramedArray[i - 1][j];
                int c = zeroFramedArray[i - 1][j + 1];
                int d = zeroFramedArray[i][j - 1];
                if (a == b == c == d == 0)
                {
```

```
                newLabel++;
                zeroFramedArray[i][j] = newLabel;
                EQArray[newLabel] = newLabel;
            }
            else
            {
                int g = 0;

                if (a != 0)
                {
                    NonZeroNeighborArray[g++] = a;
                }
                if (b != 0)
                {
                    NonZeroNeighborArray[g++] = b;
                }
                if (c != 0)
                {
                    NonZeroNeighborArray[g++] = c;
                }
                if (d != 0)
                {
                    NonZeroNeighborArray[g++] = d;
                }
```

```
int unique = findVal(NonZeroNeighborArray, g);
cout << unique << endl;
if (unique > 1)
{
    zeroFramedArray[i][j] = findMin(NonZeroNeighborArray, g);
    EQArray[newLabel++] = zeroFramedArray[i][j];
}
else
{
    if (a != 0 && (a == b || a == d))
    {
        zeroFramedArray[i][j] = a;
    }
    else if (a != 0 && a == c)
    {
        zeroFramedArray[i][j] = a;
    }
    else if (b != 0 && (b == d || b == c))
    {
        zeroFramedArray[i][j] = b;
    }
    else if (c != 0 && c == d)
    {
```

```
                            zeroFramedArray[i][j] = c;

                        }

                    }

                }

            }

        }

    }

}


void connected4pass1()

{

    for (int i = 1; i < rows + 1; i++)

    {

        for (int j = 1; j < cols + 1; j++)

        {

            if (zeroFramedArray[i][j] > 0)

            {



                int a = zeroFramedArray[i - 1][j];

                int b = zeroFramedArray[i][j - 1];

            }

        }

    }

}
```

```
void connected8pass2()

{

    for (int i = rows; i >= 1; i--)

    {

        for (int j = cols; j >= 1; j--)

        {

            if (zeroFramedArray[i][j] > 0)

            {

                int e = zeroFramedArray[i][j + 1];

                int f = zeroFramedArray[i + 1][j - 1];

                int g = zeroFramedArray[i + 1][j];

                int h = zeroFramedArray[i + 1][j + 1];



                int tmp = 0;

                if (e == f == g == h == 0)

                {

                    // nothing

                }

                else

                {

                    int v = 0;

                    if (zeroFramedArray[i][j] != 0)

                    {
                            NonZeroNeighborArray[v++] = zeroFramedArray[i][j];
                        }


                        if (e != 0)
                        {
                            NonZeroNeighborArray[v++] = e;
                        }
                        if (f != 0)
                        {
                            NonZeroNeighborArray[v++] = f;
                        }
                        if (g != 0)
                        {
                            NonZeroNeighborArray[v++] = g;
                        }
                        if (h != 0)
                        {
                            NonZeroNeighborArray[v++] = h;
                        }


                        int unique = findVal(NonZeroNeighborArray, v);
                        if (unique > 1)
                        {
                            int minLabel = findMin(NonZeroNeighborArray, v);
                            if (zeroFramedArray[i][j] > minLabel)
                            {
                                EQArray[zeroFramedArray[i][j]] = minLabel;
                            }
                            zeroFramedArray[i][j] = minLabel;
                        }
                    }
                }
            }
        }
    }
};


int main(int argc, char **argv)
{
    ifstream inFile(argv[1]);
    int rows, cols, minVal, maxVal;
    inFile >> rows >> cols >> minVal >> maxVal;
    string connectness = argv[2];
    string conversion = argv[3];
    ofstream RFprettyPrintFile(argv[4]);
    ofstream labelFile(argv[5]);
    ofstream propertyFile(argv[6]);
    ccLabel cclabel(rows, cols, minVal, maxVal);
```

```
cclabel.zero2D(cclabel.zeroFramedArray, rows + 2, cols + 2);

cclabel.loadImage(inFile, cclabel.zeroFramedArray);

cout << connectness << " " << conversion << endl;

if (conversion == "y" || conversion == "Y")

{

    cclabel.conversion();

}



// if (connectness == "4")

// {

//     cclabel.connected4pass1(RFprettyPrintFile);

// }

if (connectness == "8")

{

    cclabel.connected8pass1();

    cclabel.connected8pass2();

}



for (int i = 0; i < rows + 2; i++)

{

    for (int j = 0; j < cols + 2; j++)

    {

        cout << cclabel.zeroFramedArray[i][j] << " ";

    }
```

```
        cout << endl;

    }

    int ar[] = {1, 1, 1, 4, 5, 6, 7, 10, 10};

    int pr = cclabel.findVal(ar, 9);

    cout << pr << " " << endl;



    return 0;

}
```