CV Project 4: Implementation of the Four Basic Morphology Operations Java

Student: Michael Grossman

Project Due Date: 3/10/2022

Algorithm Steps for ComputeDilation given an input 2D Array, an output 2D array of the same size, the size of the frame, and the size of the arrays:

0. i ← rowSizeFrame
1. j← colSizeFrame
2. if input[i][j] > 0:
3. onePixelDilation(i, j, input, output)
4. end-if
5. j++
6. repeat 2 – 5 while j < colSize – colFrameSize
7. i++
8. repeat 2 -7 while i < rowSize – rowFrameSize

Algorithm Steps for ComputeErosion given an input 2D Array, an output 2D array of the same size, the size of the frame, and the size of the arrays:

0. i ← rowSizeFrame
1. j← colSizeFrame
2. if input[i][j] > 0:
3. onePixelErosion(i, j, input, output)
4. end-if
5. j++
6. repeat 2 – 5 while j < colSize – colFrameSize
7. i++
8. repeat 2 -7 while i < rowSize – rowFrameSize

Algorithm Steps for OnePixelErosion given a pixel location (i,j), an input 2D Array, an output 2D array, a Structured Element 2D array, and an origin location for the Structured Element:

1. iOffset ←i – rowOrigin
2. jOffest ←j – colOrigin
3. matchFlag ← true
4. r ← 0
5. c ← 0
6. if (struct[r][c] > 0) and (input[iOffset + r][jOffset + c]) <= 0)
7. matchFlag ← false
8. End-if
9. c++
10. repeat 6 – 9 while (matchFlag == true) and (c < numStructCols)
11. r++
12. repeat 6 – 11 while (matchFlag == true) and (r < numStructRows)

Algorithm Steps for ComputeClosing given an input 2D array, a temp array of the same size, an output array, and all variables required for erosion and dilation:

0. ComputeDilation(input, temp)
1. ComputeErosion(temp, output)

Algorithm Steps for ComputeOpening given an input 2D array, a temp array of the same size, an output array, and all variables required for erosion and dilation:

0. ComputeErosion(input, temp)
1. ComputeDilation(temp, output)

Main.java

```
import java.io.*;
import java.util.Scanner;


public class Main {
    public static void main(String[] args) {
        Scanner imgFile, structFile;
        BufferedWriter dialateOutFile, erodeOutFile, closingOutFile;
        BufferedWriter openingOutFile, prettyPrintFile;
        Morphology morphology;
        try{
            imgFile = new Scanner(new BufferedReader(new FileReader(args[0])));
            structFile = new Scanner(new BufferedReader(
                new FileReader(args[1])));
            dialateOutFile = new BufferedWriter(new FileWriter(args[2]));
            erodeOutFile = new BufferedWriter(new FileWriter(args[3]));
            closingOutFile = new BufferedWriter(new FileWriter(args[4]));
            openingOutFile = new BufferedWriter(new FileWriter(args[5]));
            prettyPrintFile = new BufferedWriter( new FileWriter(args[6]));

            //Constructor loads file info into the object
            morphology = new Morphology(imgFile, structFile);

            prettyPrintFile.write("Initial Zero Framed 2D Array: \n");
            Morphology.prettyPrint(morphology.zeroFramedAry, prettyPrintFile);
            prettyPrintFile.write("Initial Structured Element 2D Array: \n");
            Morphology.prettyPrint(morphology.structAry, prettyPrintFile);

            Morphology.zero2DAry(morphology.morphAry, morphology.rowSize,
                                 morphology.colSize);
            morphology.computeDilation(morphology.zeroFramedAry,
                                       morphology.morphAry);
            Morphology.aryToFile(morphology.morphAry, dialateOutFile);
            prettyPrintFile.write("Dilation Operation Output: \n");
            Morphology.prettyPrint(morphology.morphAry, prettyPrintFile);

            Morphology.zero2DAry(morphology.morphAry, morphology.rowSize,
                                 morphology.colSize);
            morphology.computeErosion(morphology.zeroFramedAry,
                                      morphology.morphAry);
            Morphology.aryToFile(morphology.morphAry, erodeOutFile);
            prettyPrintFile.write("Erosion Operation Output: \n");
```

```
            Morphology.prettyPrint(morphology.morphAry, prettyPrintFile);

            Morphology.zero2DAry(morphology.morphAry, morphology.rowSize,
                                 morphology.colSize);
            morphology.computeOpening(morphology.zeroFramedAry,
                                      morphology.morphAry,
                                      morphology.tempAry);
            Morphology.aryToFile(morphology.morphAry, openingOutFile);
            prettyPrintFile.write("Opening Operation Output:\n");
            Morphology.prettyPrint(morphology.morphAry, prettyPrintFile);

            Morphology.zero2DAry(morphology.morphAry, morphology.rowSize,
                                 morphology.colSize);
            morphology.computeClosing(morphology.zeroFramedAry,
                                      morphology.morphAry,
                                      morphology.tempAry);
            Morphology.aryToFile(morphology.morphAry, closingOutFile);
            prettyPrintFile.write("Closing Operation Output: \n");
            Morphology.prettyPrint(morphology.morphAry, prettyPrintFile);


            imgFile.close();
            structFile.close();
            dialateOutFile.close();
            erodeOutFile.close();
            closingOutFile.close();
            openingOutFile.close();
            prettyPrintFile.close();


        }catch(FileNotFoundException e){
            System.out.println(e.getMessage());
            System.exit(1);
        }
        catch(IOException e){
            System.out.println(e.getMessage());
            System.exit(1);
        }

    }
}
```

Morphology.java

```java
import java.io.*;
import java.util.Scanner;

public class Morphology {
    public int numImgRows, numImgCols, imgMin, imgMax;
    public int numStructRows, numStructCols, structMin, structMax;
    public int rowOrigin, colOrigin;
    public int rowFrameSize, colFrameSize;
    public int extraRows, extraCols, rowSize, colSize;
    public int[][] zeroFramedAry, morphAry, tempAry, structAry;

    public Morphology(Scanner img, Scanner struct){
        numImgRows = img.nextInt();
        numImgCols = img.nextInt();
        imgMin = img.nextInt();
        imgMax = img.nextInt();

        numStructRows = struct.nextInt();
        numStructCols = struct.nextInt();
        structMin = struct.nextInt();
        structMax = struct.nextInt();
        rowOrigin = struct.nextInt();
        colOrigin = struct.nextInt();

        rowFrameSize = numStructRows / 2;
        colFrameSize = numStructCols / 2;

        extraRows = rowFrameSize*2;
        extraCols = colFrameSize*2;

        rowSize = numImgRows + extraRows;
        colSize = numImgCols + extraCols;

        zeroFramedAry = new int[rowSize][colSize];
        morphAry = new int[rowSize][colSize];
        tempAry = new int[rowSize][colSize];
        structAry = new int[numStructRows][numStructCols];

        Morphology.zero2DAry(zeroFramedAry, rowSize, colSize);
        loadImg(img, zeroFramedAry);

        Morphology.zero2DAry(structAry, numStructRows, numStructCols);
        loadStruct(struct, structAry);
```

```
    }

    public static void zero2DAry(int[][] ary, int nRows, int nCols){
        for(int i = 0; i < nRows; ++i){
            for(int j = 0; j < nCols; ++j){
                ary[i][j] = 0;
            }
        }
    }


    private void loadImg(Scanner imgFile, int[][] zeroFramedArray){
        int rowEnd = rowSize - rowFrameSize, colEnd = colSize - colFrameSize;
        for(int i = rowFrameSize; i < rowEnd; ++i){
            for(int j = colFrameSize; j < colEnd; ++j){
                zeroFramedArray[i][j] = imgFile.nextInt();
            }
        }
    }

    private void loadStruct(Scanner structFile, int[][] structArray){
        for(int i = 0; i < numStructRows; ++i){
            for(int j = 0; j < numStructCols; ++j){
                structArray[i][j] = structFile.nextInt();
            }
        }
    }

    public void computeDilation(int[][] inAry, int[][] outAry){
        for(int i = rowFrameSize; i < rowSize - rowFrameSize; ++i){
            for(int j = colFrameSize; j < colSize-colFrameSize; ++j){
                if(inAry[i][j] > 0){
                    //do we need the inAry? seems irrelevent
                    onePixelDilation(i, j, inAry, outAry);
                }
            }
        }

    }

    public void computeErosion(int[][] inAry, int[][] outAry){
        for(int i = rowFrameSize; i < rowSize - rowFrameSize; ++i){
            for(int j = colFrameSize; j < colSize - colFrameSize; ++j){
                if(inAry[i][j] > 0){
                    onePixelErosion(i, j, inAry, outAry);
```

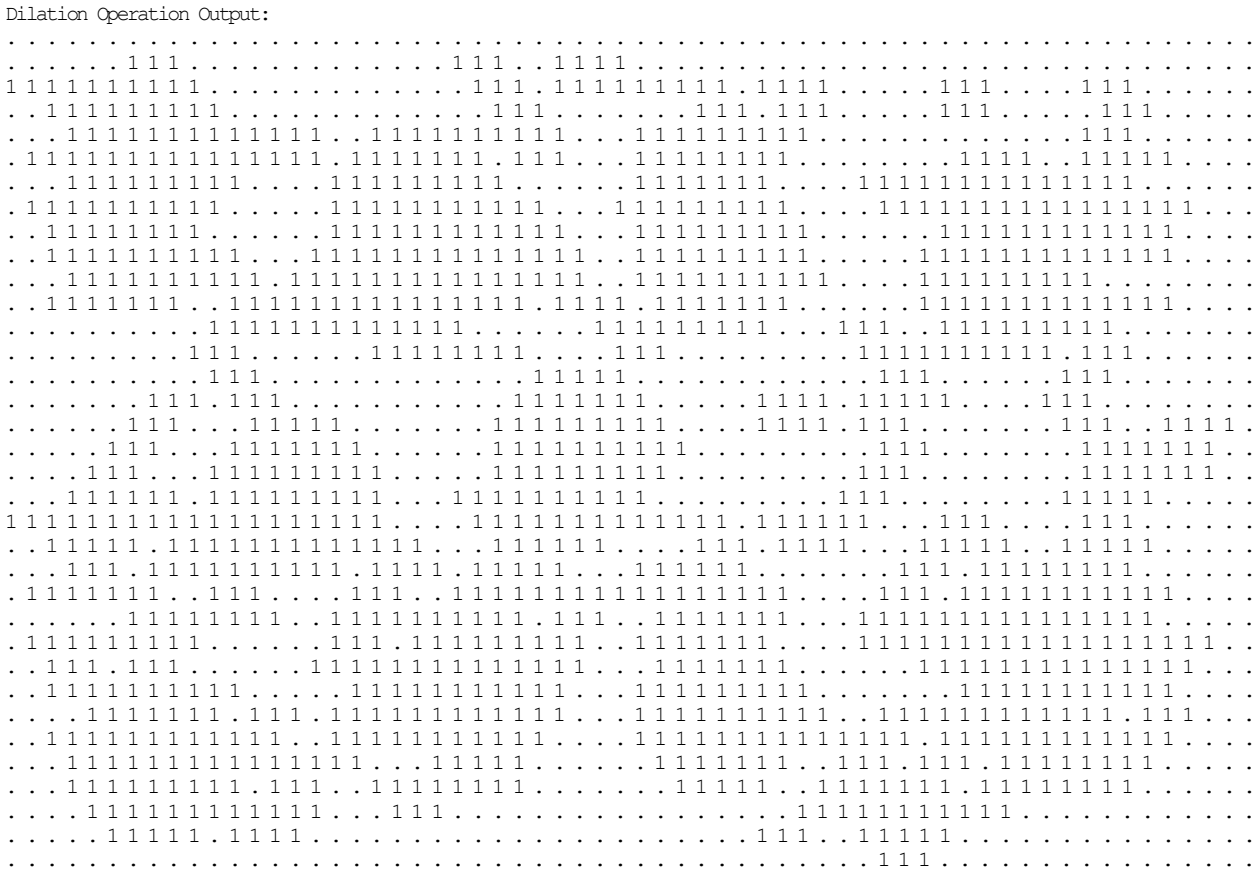
```
                }
            }
        }
    }

    public void computeOpening(int[][] inAry, int[][] outAry, int [][] temp){
        computeDilation(inAry, tempAry);
        computeErosion(tempAry, outAry);
    }

    public void computeClosing(int[][] inAry, int[][] outAry, int [][] temp){
        computeErosion(inAry, tempAry);
        computeDilation(tempAry, outAry);
    }

    public void onePixelDilation(int i, int j, int[][] inAry, int[][] outAry){
        int iOffset = i - rowOrigin, jOffset = j - colOrigin;

        for(int r = 0; r < numStructRows; ++r){
            for(int c = 0; c< numStructCols; ++c){
                if(structAry[r][c] > 0){
                    outAry[iOffset + r][jOffset + c] = 1;
                }
            }
        }
    }

    public void onePixelErosion(int i, int j, int[][] inAry, int[][] outAry){
        boolean matchFlag = true;
        int iOffset = i - rowOrigin, jOffset = j - colOrigin;

        for(int r = 0; r < numStructRows && matchFlag; ++r){
            for(int c = 0; c < numStructCols && matchFlag; ++c){
                if(structAry[r][c] > 0 && inAry[iOffset + r][jOffset + c] <= 0){
                    matchFlag = false;
                }
            }
        }
        if(matchFlag){
            outAry[i][j] = 1;
        }
        else{
            outAry[i][j] = 0;
        }
    }
```

```
    public static void aryToFile(int[][] ary, BufferedWriter outfile){
        try{
            int r = ary.length, c = 0;
            for(int i = 0; i < r; ++i){
                c = ary[i].length;
                for(int j = 0; j < c; ++j){
                    outfile.write(Integer.toString(ary[i][j]) + " ");
                }
                outfile.write("\n");
            }
            outfile.write("\n\n");
        }
        catch(IOException e){
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void prettyPrint(int[][] ary, BufferedWriter outfile){
        try{
            int r = ary.length, c = 0;
            for(int i = 0; i < r; ++i){
                c = ary[i].length;
                for(int j = 0; j < c; ++j){
                    if(ary[i][j] == 0){
                        outfile.write(". ");
                    }
                    else{
                        outfile.write("1 ");
                    }
                }
                outfile.write("\n");
            }
            outfile.write("\n\n");
        }
        catch(IOException e){
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

}
```

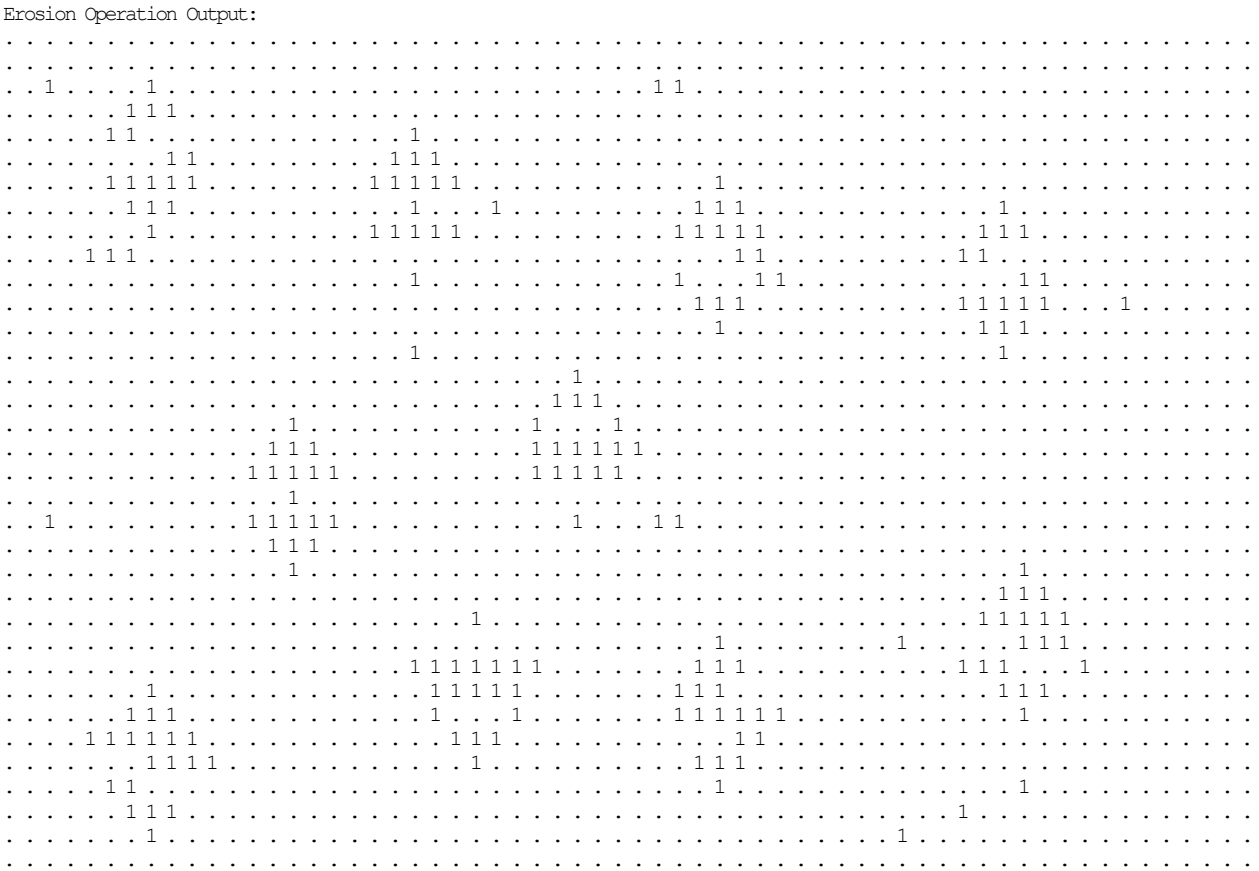
OUTPUT

```
Image 1, Structured Element 1
**************************************
Initial Zero Framed 2D Array:
. . . . . .
. . . . . 1 . . . . . 1 . . . . 1 1 . . . . . .
. 1 1 1 . . 1 1 1 . . . . . 1 . . . 1 1 . 1 1 1 1 . . . 1 1 . . . . . 1 . . . . . 1 . . . . . .
. . . 1 . 1 1 1 1 1 . . . . . 1 . . . . . 1 . . . 1 . . . . . 1 . . . . . 1 . . . . . .
. . . . 1 1 1 1 . 1 1 . . 1 1 . . . . 1 1 1 . . 1 . 1 . . . . . 1 . 1 . . 1 1 . . . . . 1 . . . . . .
. . 1 . 1 1 . 1 1 1 1 . . 1 1 . . . 1 1 1 1 1 . . . 1 . . . . . 1 . . 1 . 1 . . . . . 1 1 . . . . 1 . 1 . . . . .
. . . . 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 . . . . . 1 . 1 1 1 . . . . . 1 . . 1 1 . 1 . 1 1 . 1 . . . . . .
. . 1 . . 1 1 1 1 1 . . . . . 1 . 1 1 1 . 1 1 1 . . . . . 1 . 1 1 1 1 1 . . . . . 1 . 1 . 1 1 1 . 1 . 1 1 . 1 . . . .
. . . 1 . . 1 1 1 . . . . . 1 1 1 1 1 1 1 . . 1 . . . . . 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 . 1 . . 1 . . . . .
. . . 1 1 1 1 1 . 1 1 . . . . . 1 . 1 1 . 1 1 . 1 . . 1 . . . . 1 1 . 1 1 1 1 . . . . . 1 1 1 1 . 1 1 . 1 . 1 . . . . .
. . . . 1 . 1 1 . 1 . 1 . . . 1 . 1 . 1 1 1 . 1 1 . . 1 . . . . 1 1 1 . 1 1 1 1 . . . . . 1 1 . 1 1 1 1 . . . . . .
. . . 1 . 1 . 1 . . . . 1 . 1 . . 1 . . 1 . 1 . 1 . . . 1 1 . . . 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 . 1 1 1 . . . . .
. . . . . 1 . . 1 . 1 . . 1 . 1 . . . . . 1 . 1 . 1 1 1 . . . . 1 . . . . 1 1 1 1 1 . 1 . . . . . .
. . . . . 1 . . . . . 1 1 1 . . 1 . . . . . 1 . . . . . 1 . 1 . . 1 1 1 . . . 1 . . . . . .
. . . . . 1 . . . . . 1 1 1 . . . . . 1 . . . . . 1 . . . . . 1 . . . . . .
. . . . . 1 . . . 1 . . . . . 1 1 1 1 1 . . . . . 1 1 . . . 1 . 1 . . . . . 1 . . . . . .
. . . . . 1 . . . . . 1 1 1 . . . 1 1 1 . 1 1 1 . . . . . 1 1 . . . 1 . . . . . 1 . . . . 1 1 . .
. . . . . 1 . . . . . 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 . . . . . 1 . . . . . 1 . . 1 1 . . .
. . . . . 1 . . . . . 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 . . . . . 1 . . . . . 1 . . 1 1 . . .
. . . . 1 . . 1 . . . 1 . 1 1 1 . 1 . . . . . 1 . . 1 1 . 1 1 . . . . . 1 . . . . . 1 . 1 . . . . . .
. 1 1 1 . . 1 . 1 1 . 1 1 1 1 1 1 1 . . . . . 1 . . 1 1 1 . 1 1 1 1 . . . 1 1 . 1 . . . . . 1 . . . . . 1 . . . . . .
. . . 1 . 1 . . . 1 . . 1 1 1 1 1 . . 1 . . . . . 1 . 1 . . . . . 1 . . . 1 1 . . . 1 . 1 . . . . . .
. . . . 1 . . . 1 . 1 . . 1 1 1 . . . 1 1 . . . . 1 . 1 . . . . . 1 . 1 1 . . . . . 1 . . . . . .
. . 1 . . 1 1 . . . . 1 . . . . . 1 . . . . 1 . . 1 . . 1 . 1 . . 1 . 1 . . . . . 1 . . . 1 1 1 1 1 . 1 . 1 . . . . .
. . . . . 1 . 1 . . 1 . . . . 1 1 . 1 . 1 1 1 . . . 1 . . . . 1 . 1 . 1 . . . . . 1 1 . . 1 1 1 1 1 1 1 . 1 . . . . . .
. . 1 . 1 1 . 1 1 . . . . . . . . 1 . . . 1 1 . 1 1 . 1 1 1 . . . . . 1 1 1 . 1 . 1 1 1 1 1 . 1 1 . 1 . . . .
. . . 1 . . . 1 . . . . . . . . . 1 . . 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 . . . . . 1 1 1 1 1 . 1 1 1 . . 1 . . . .
. . . 1 1 . 1 1 1 . 1 . . . . . 1 . 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 . 1 . . . . . 1 1 1 1 1 . 1 . 1 . . . . .
. . . . . 1 1 1 1 1 . . . 1 . . . 1 . . 1 1 1 . 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 . . . 1 . . 1 . 1 1 1 . 1 . . . 1 . . . .
. . . 1 1 1 1 1 1 1 1 . 1 . . . . 1 . 1 . 1 1 1 1 1 . . . . . 1 1 . 1 1 1 1 . 1 1 . 1 . . . 1 . . 1 . 1 . 1 . 1 . . . . .
. . . . 1 . 1 1 1 1 1 1 . 1 . . 1 . . . . . 1 1 1 . . . . . . . 1 1 1 1 1 . . . . 1 . . . 1 . . . 1 . 1 1 . 1 . . . . . .
. . . . 1 1 1 1 . 1 1 . . . 1 . . . . 1 . . 1 . . . . . 1 . . . . . 1 1 1 . . . . 1 . 1 . 1 . . . 1 1 1 . . 1 . . . . . .
. . . . . 1 1 1 1 1 . 1 . . 1 . . . . . 1 . . . . . . . . . . . . . . . . . . . . . 1 . . 1 1 . 1 1 1 . . . . . .
. . . . . 1 1 1 . . . 1 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 . . . . 1 1 1 . . . . . .
. . . . . 1 .
Initial Structured Element 2D Array:

1 1 1
```

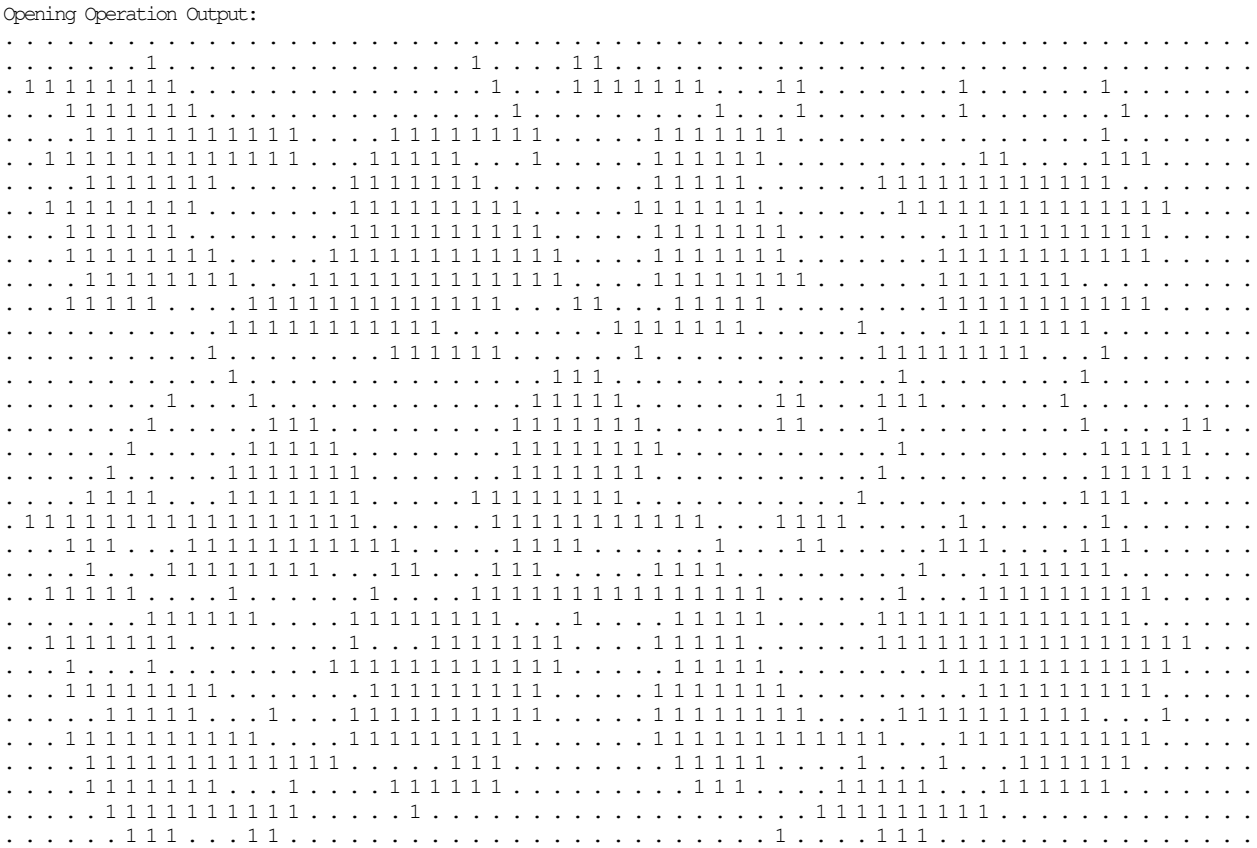
Dilation Operation Output:



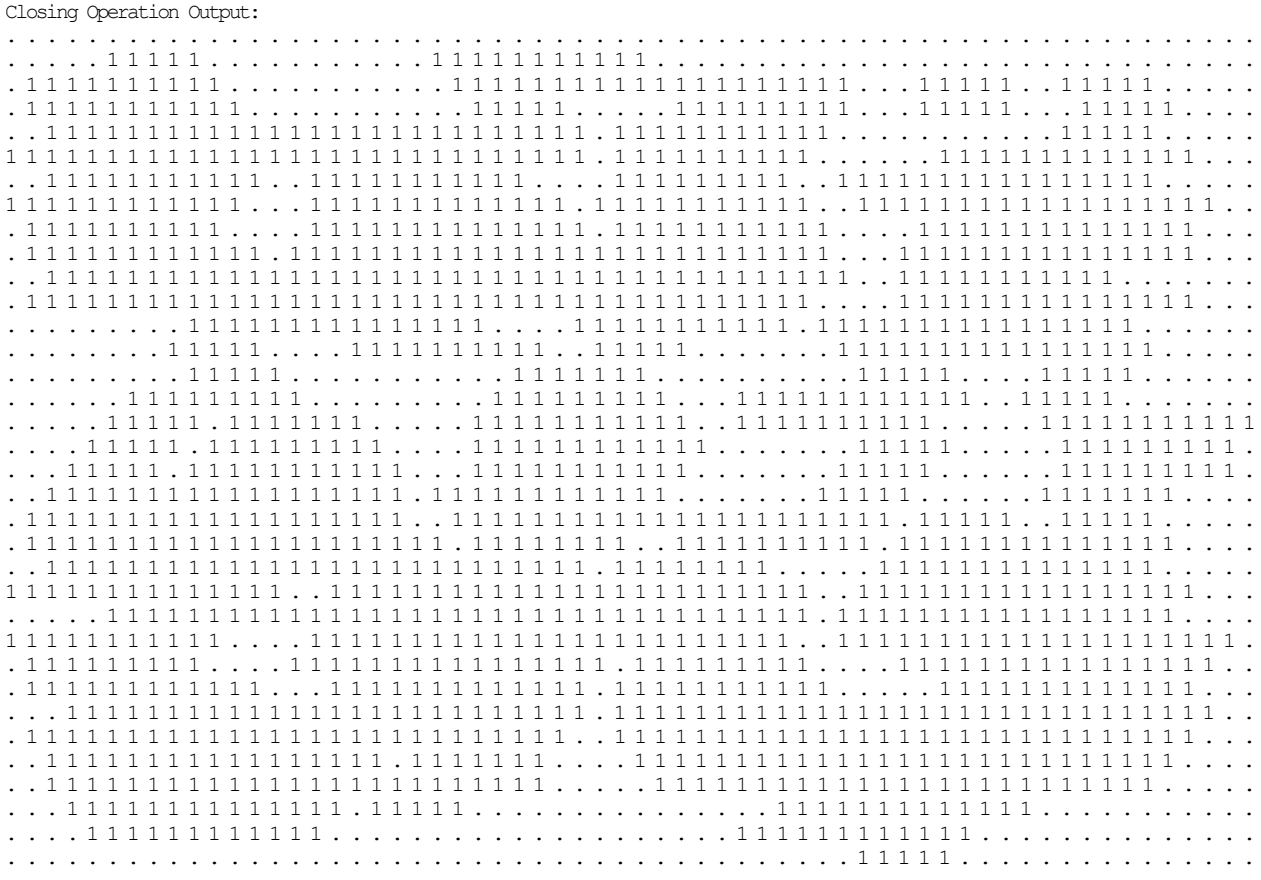
Erosion Operation Output:



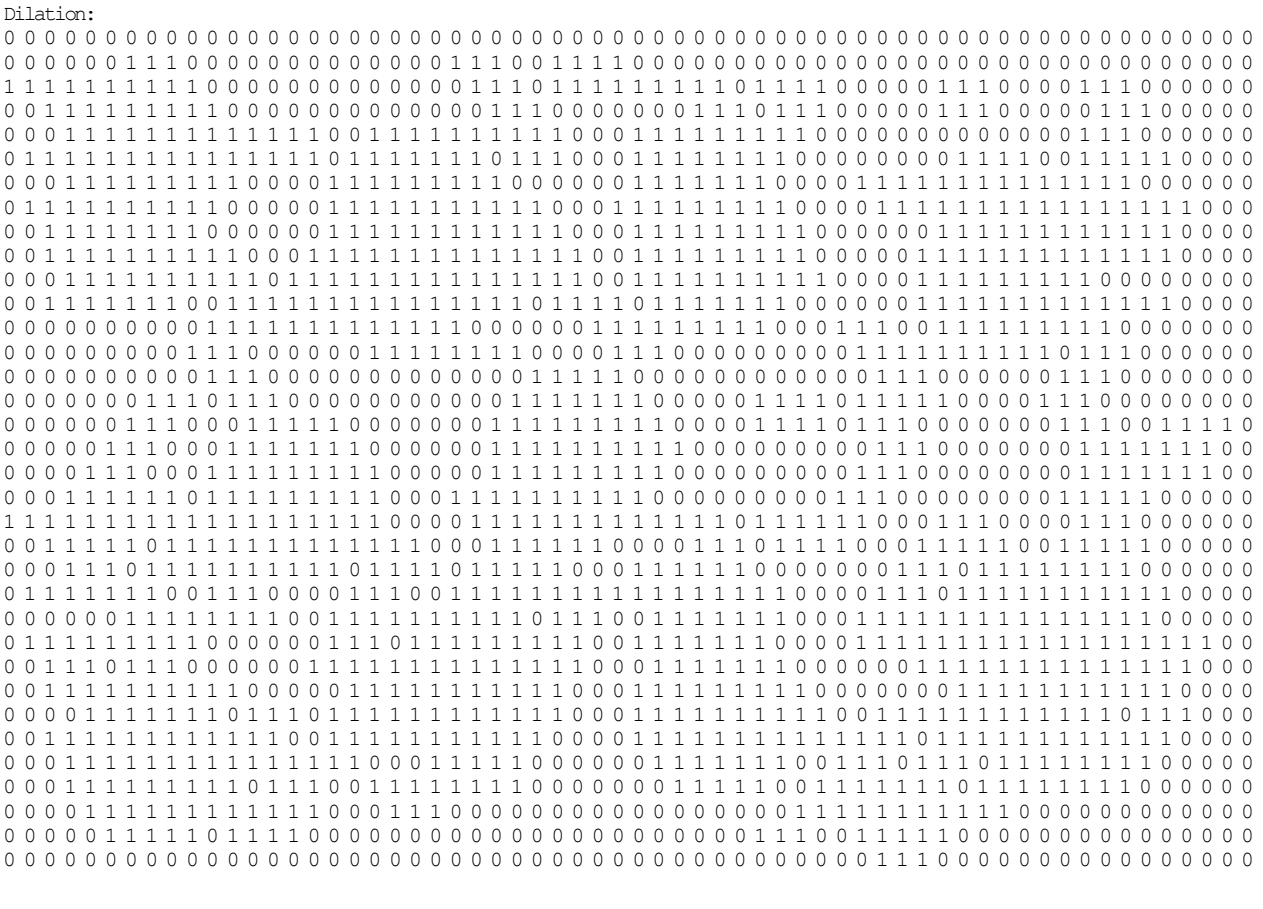
Opening Operation Output:



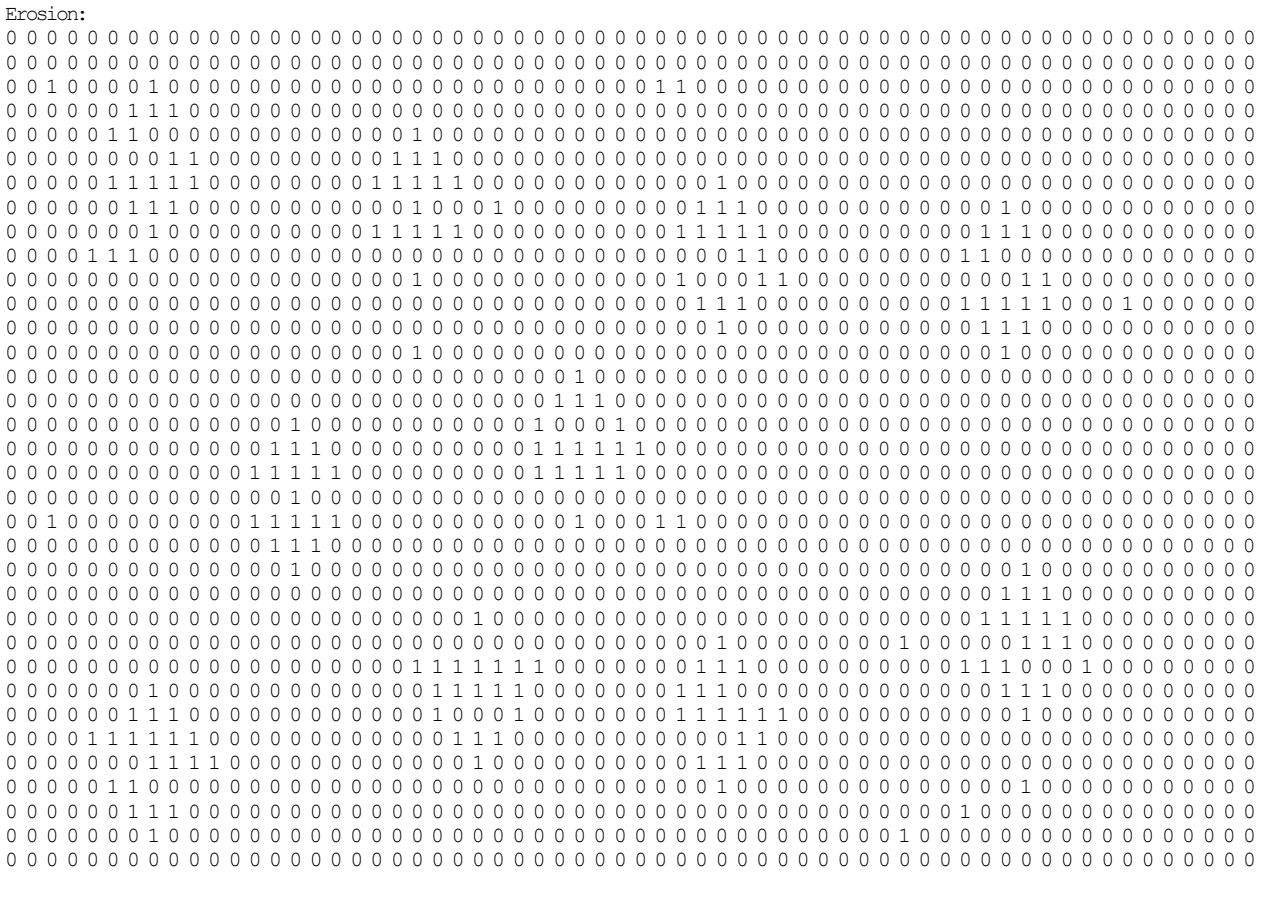
Closing Operation Output:



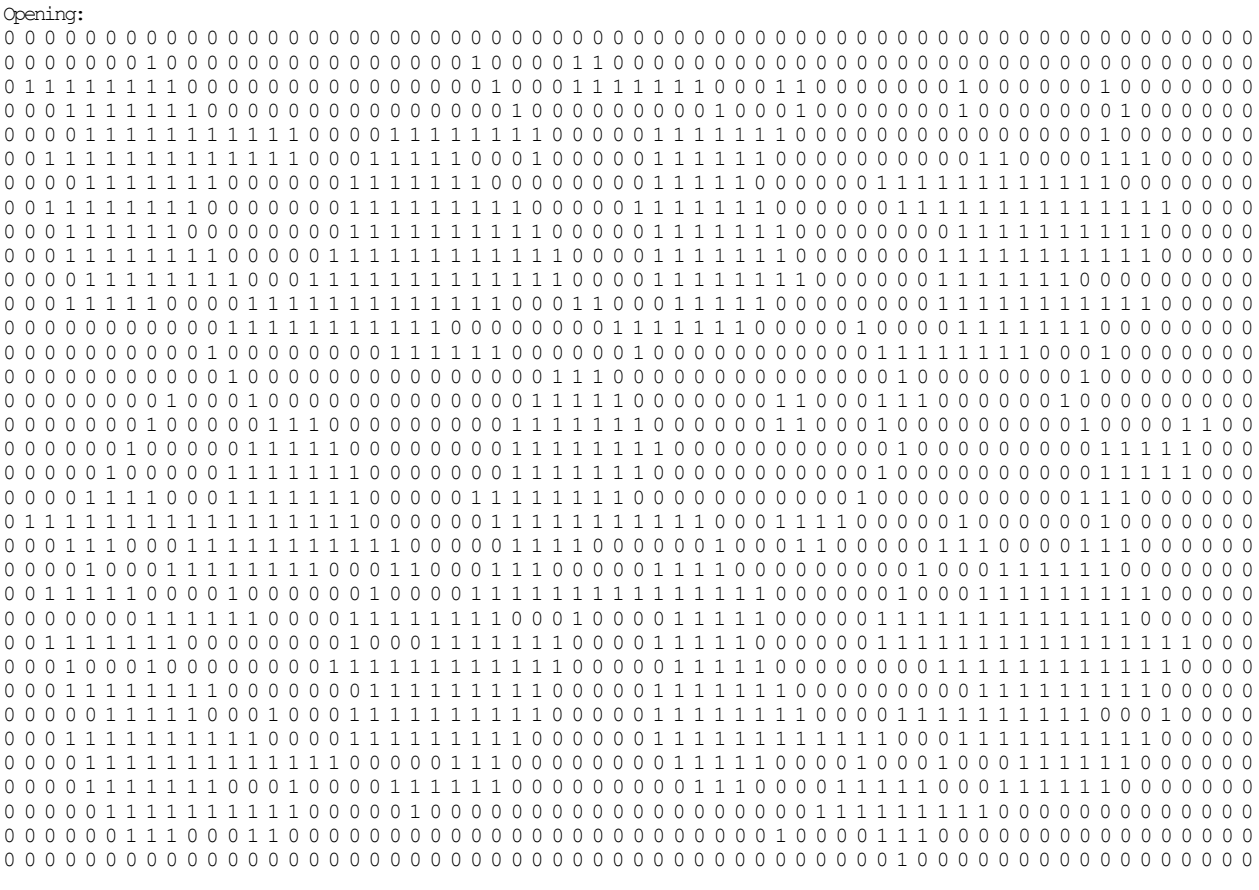
Dilation:



Erosion:



Opening:



Closing:

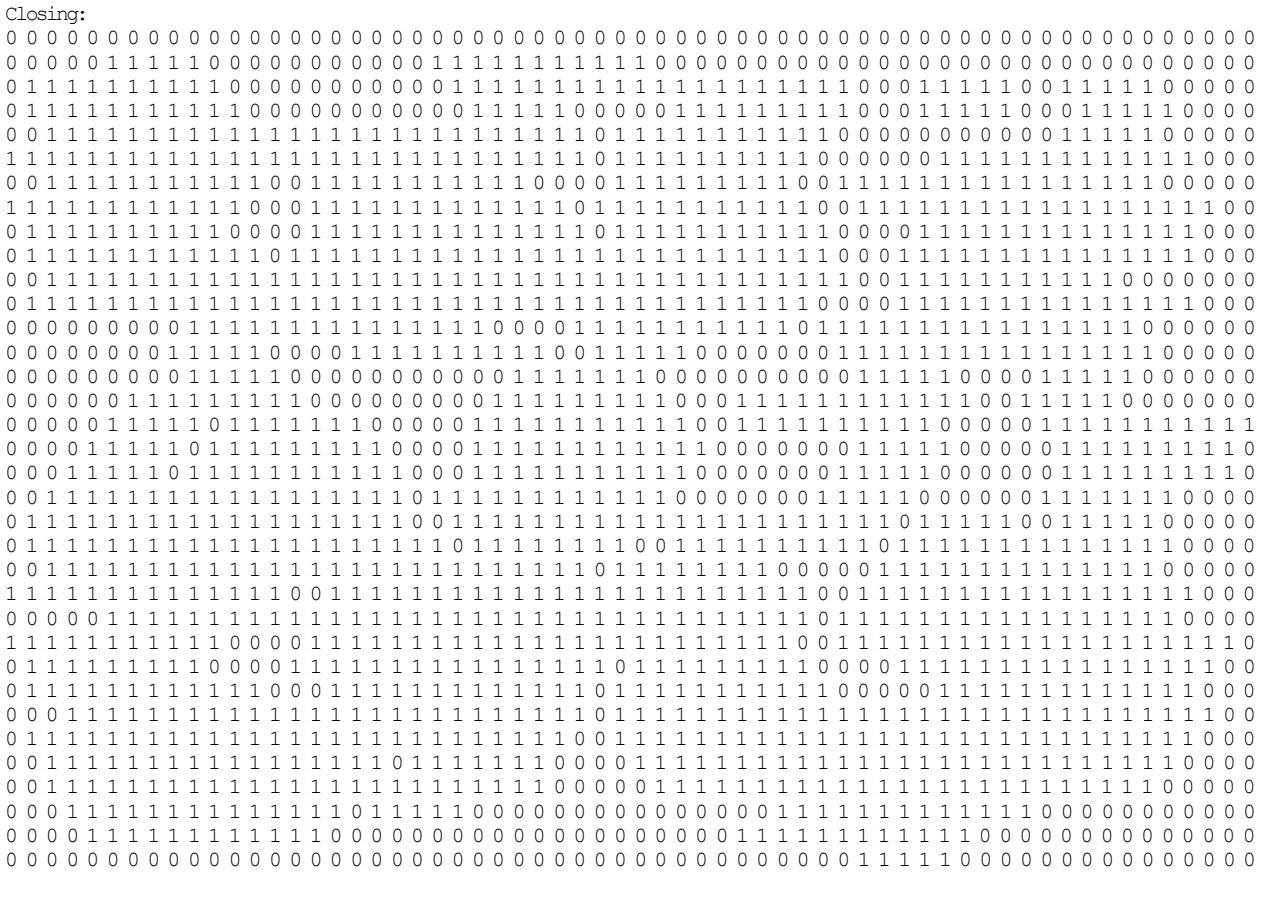
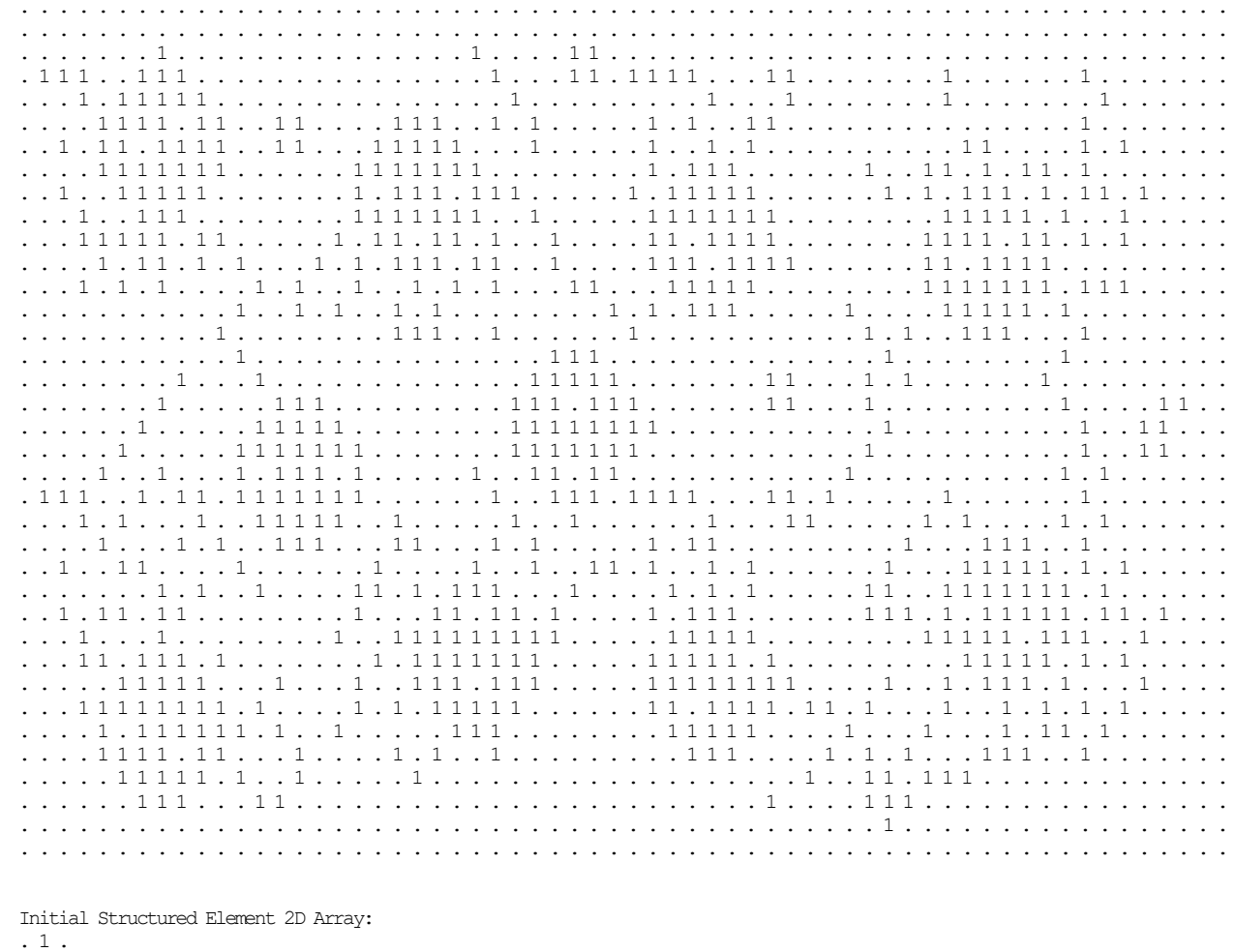


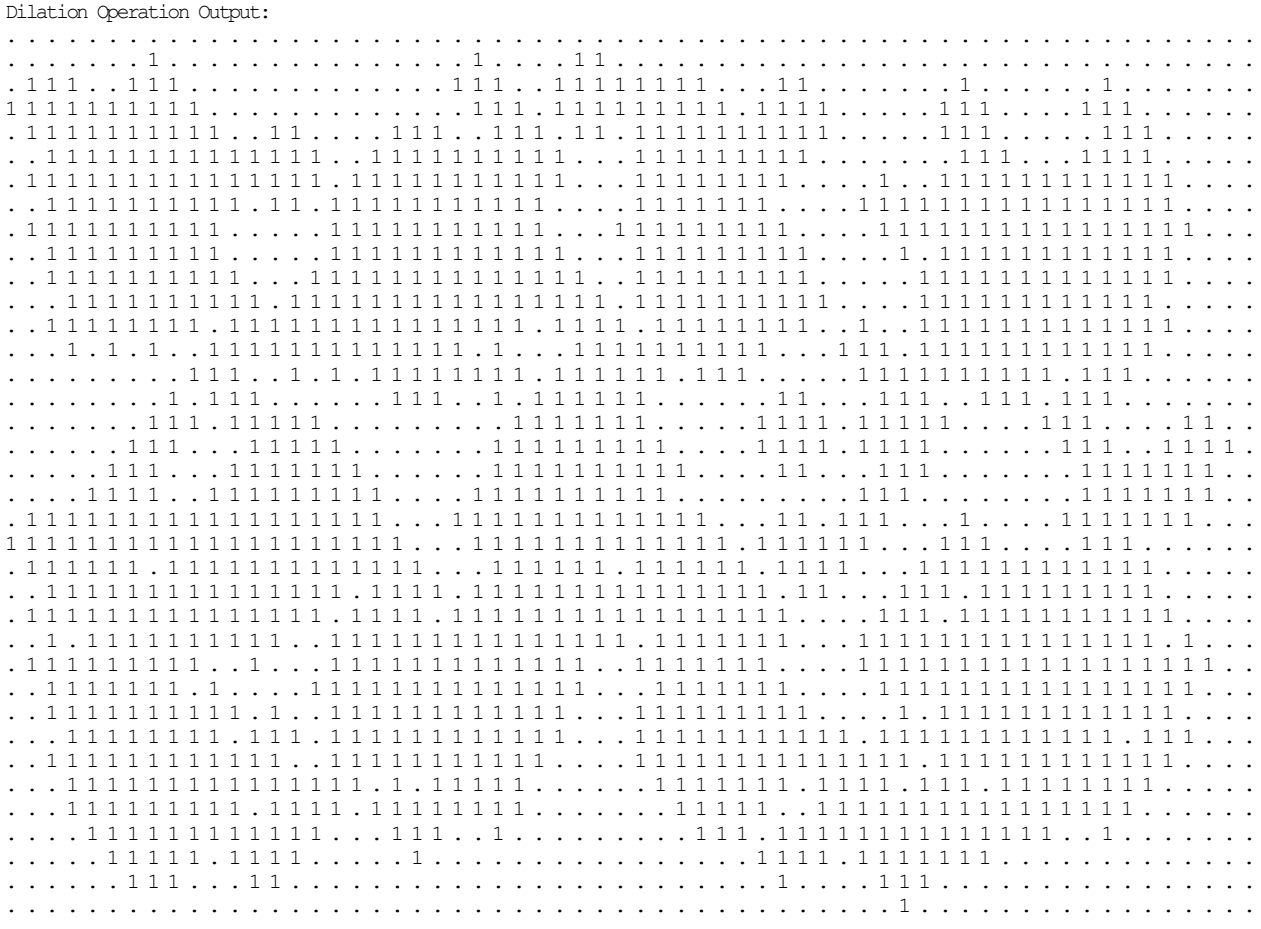
Image 1, Structured Element 2
****
Initial Zero Framed 2D Array:



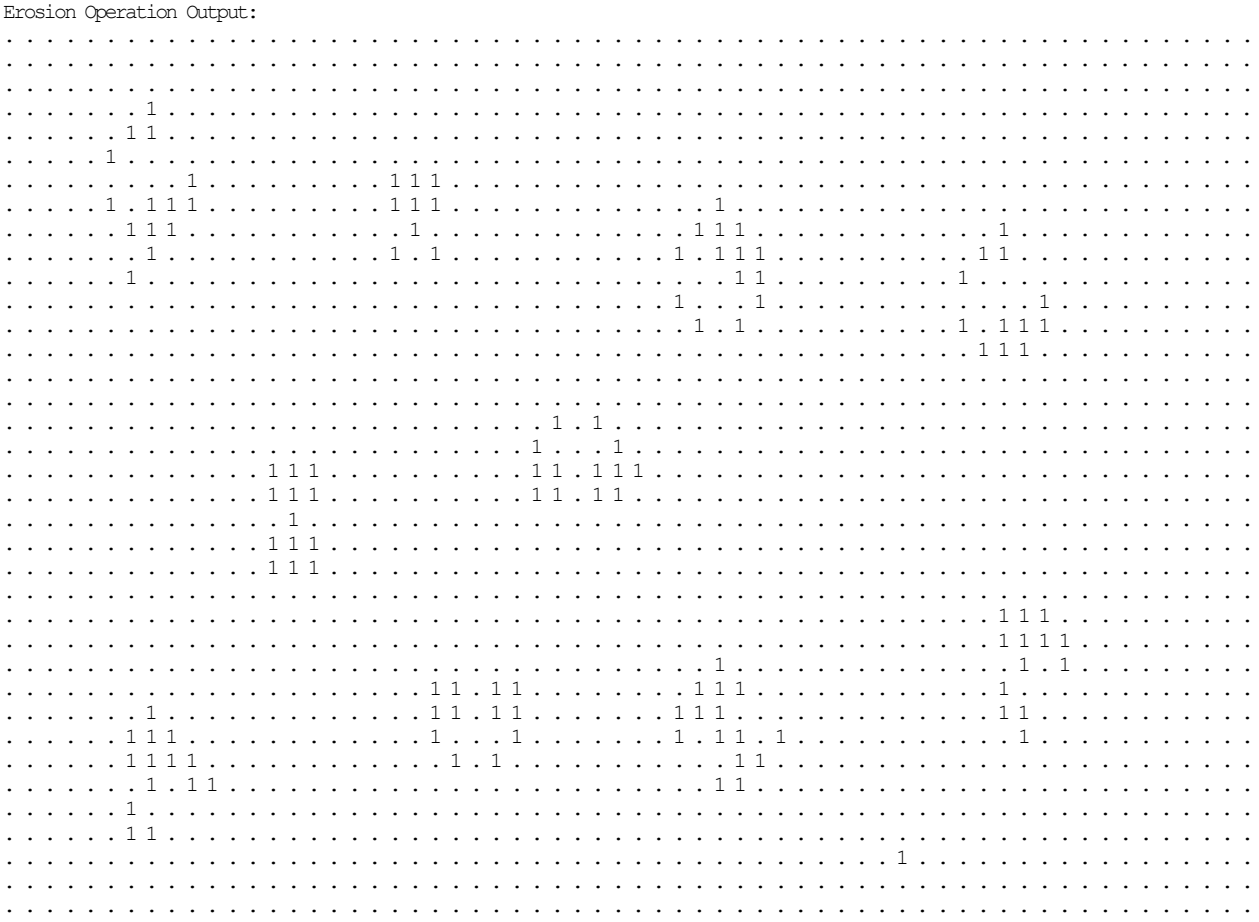
Initial Structured Element 2D Array:

```
. 1 .
1 1 1
. 1 .
```

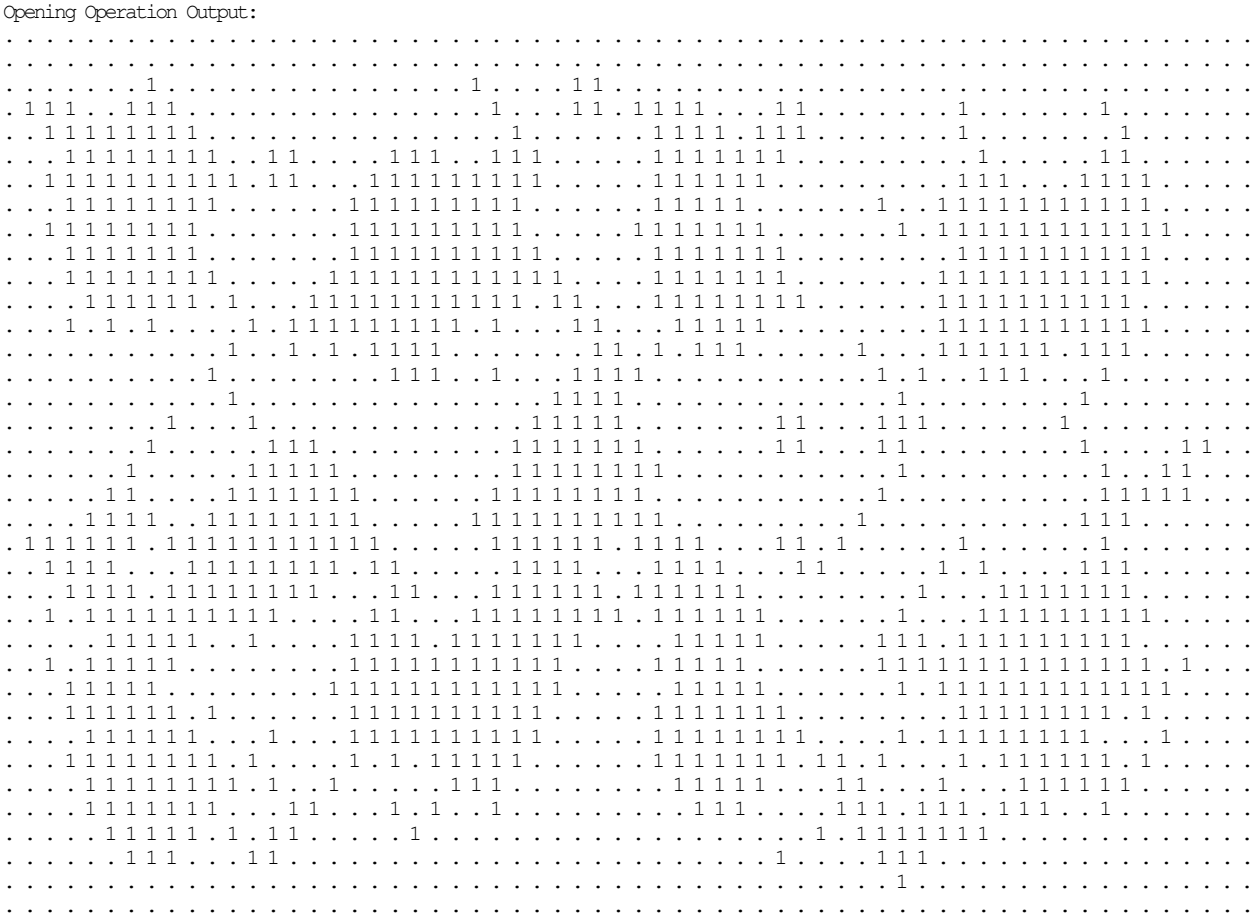
Dilation Operation Output:



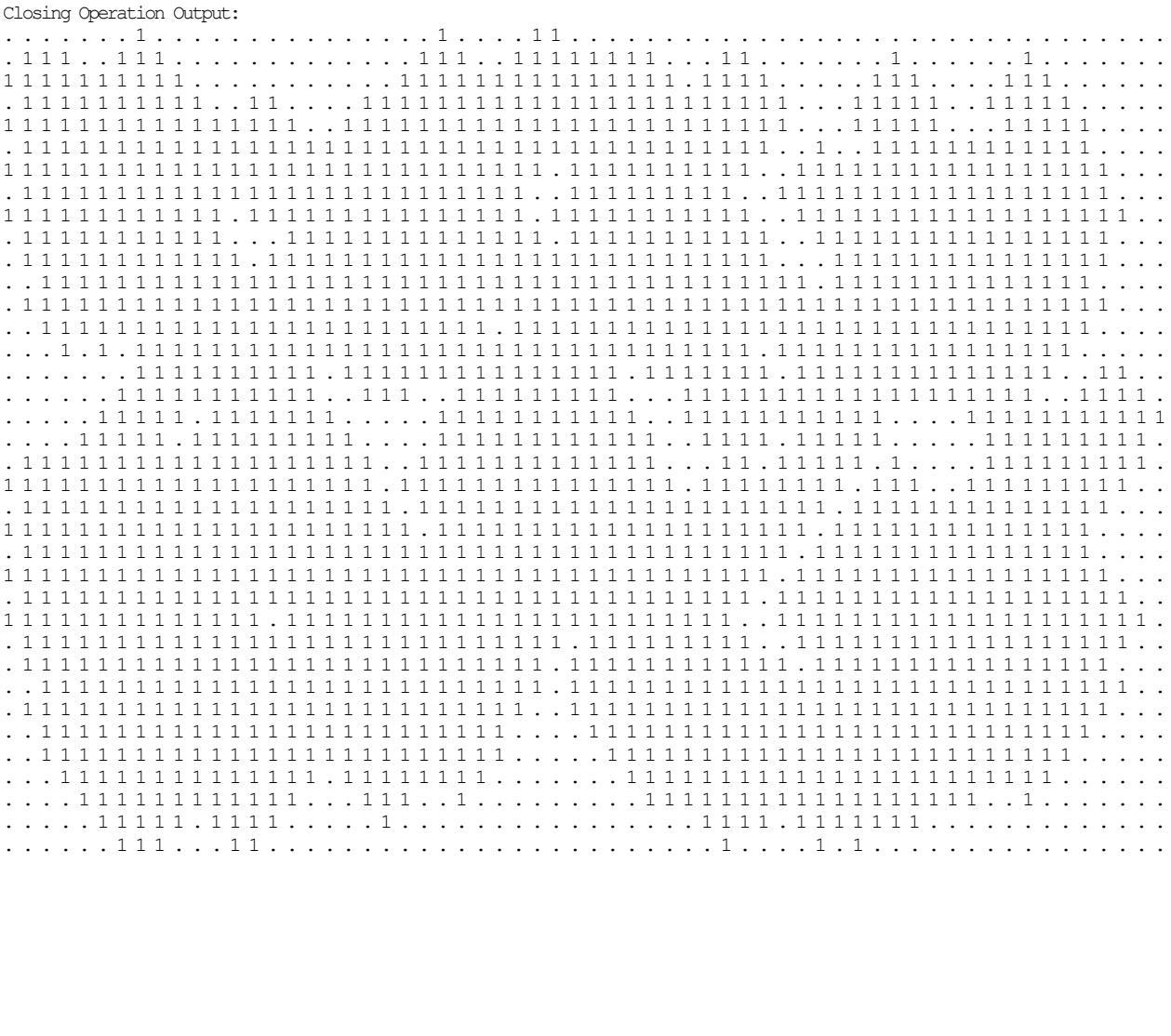
Erosion Operation Output:



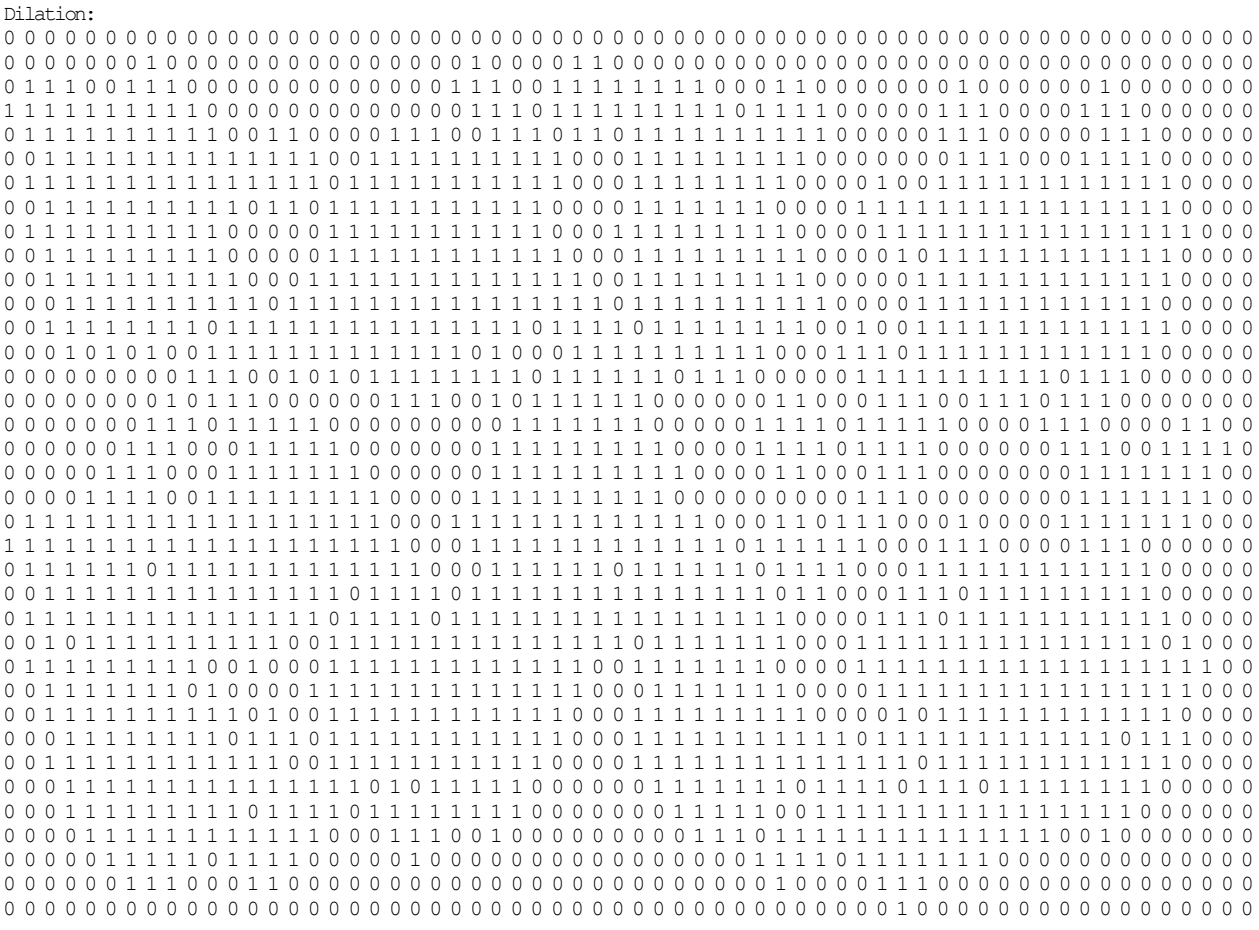
Opening Operation Output:



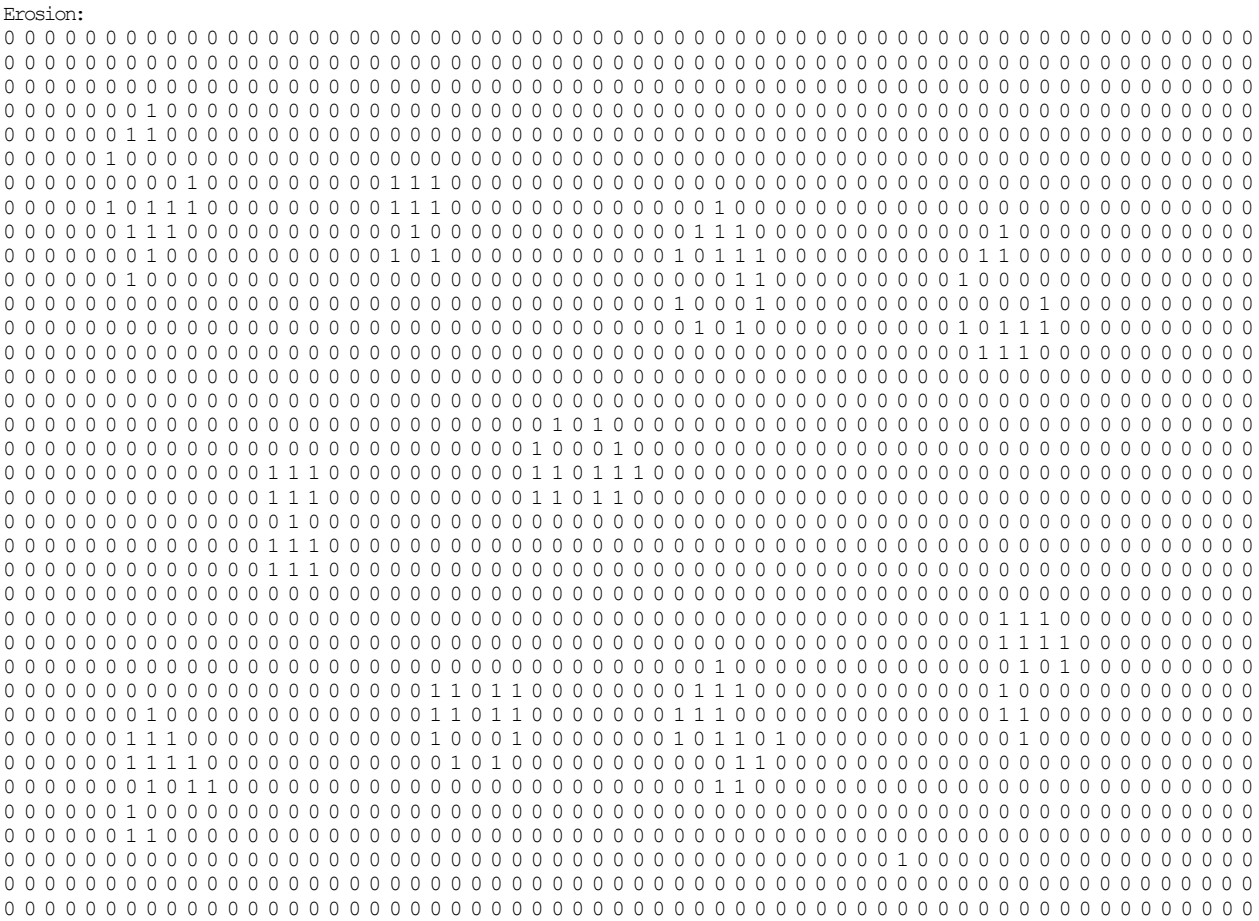
Closing Operation Output:



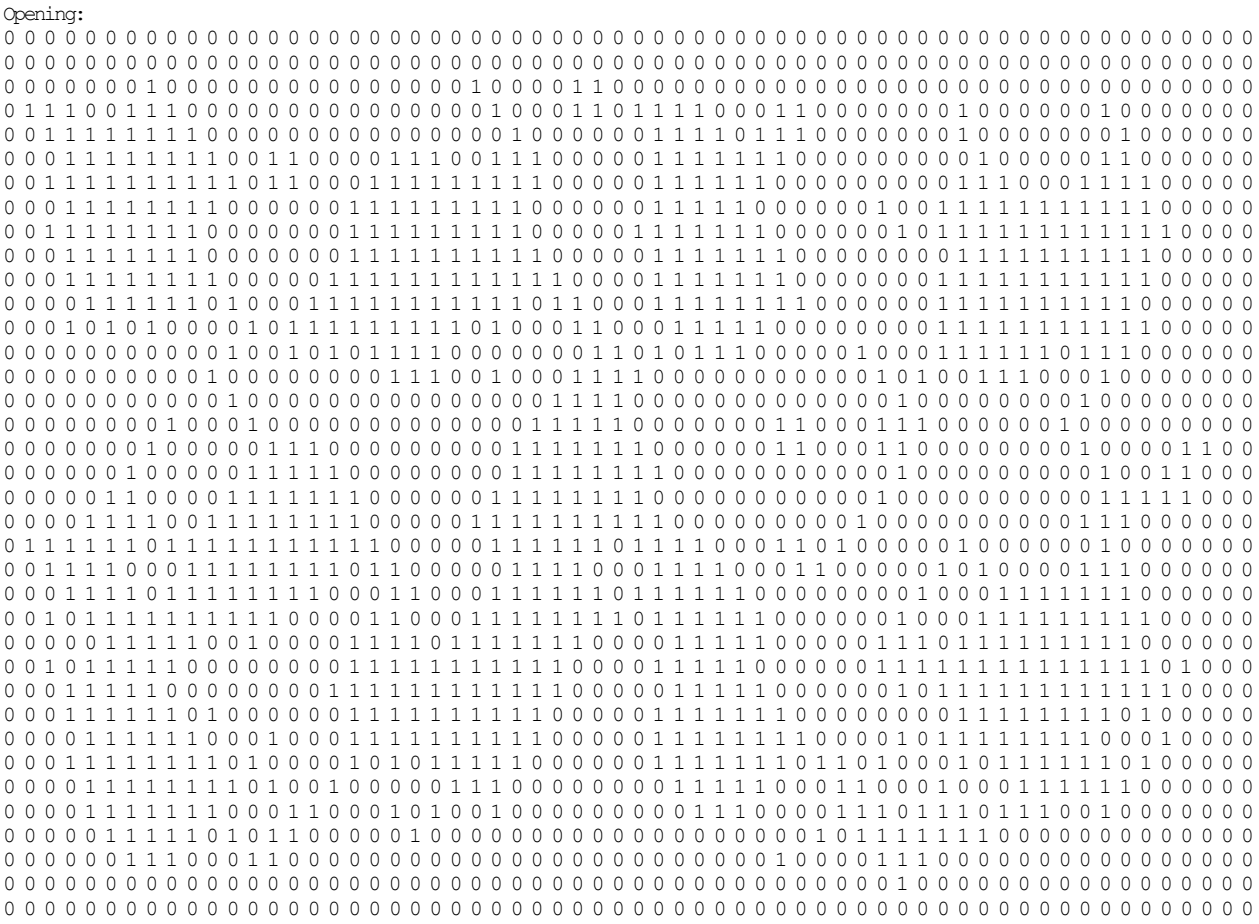
Dilation:



Erosion:



Opening:



Closing:

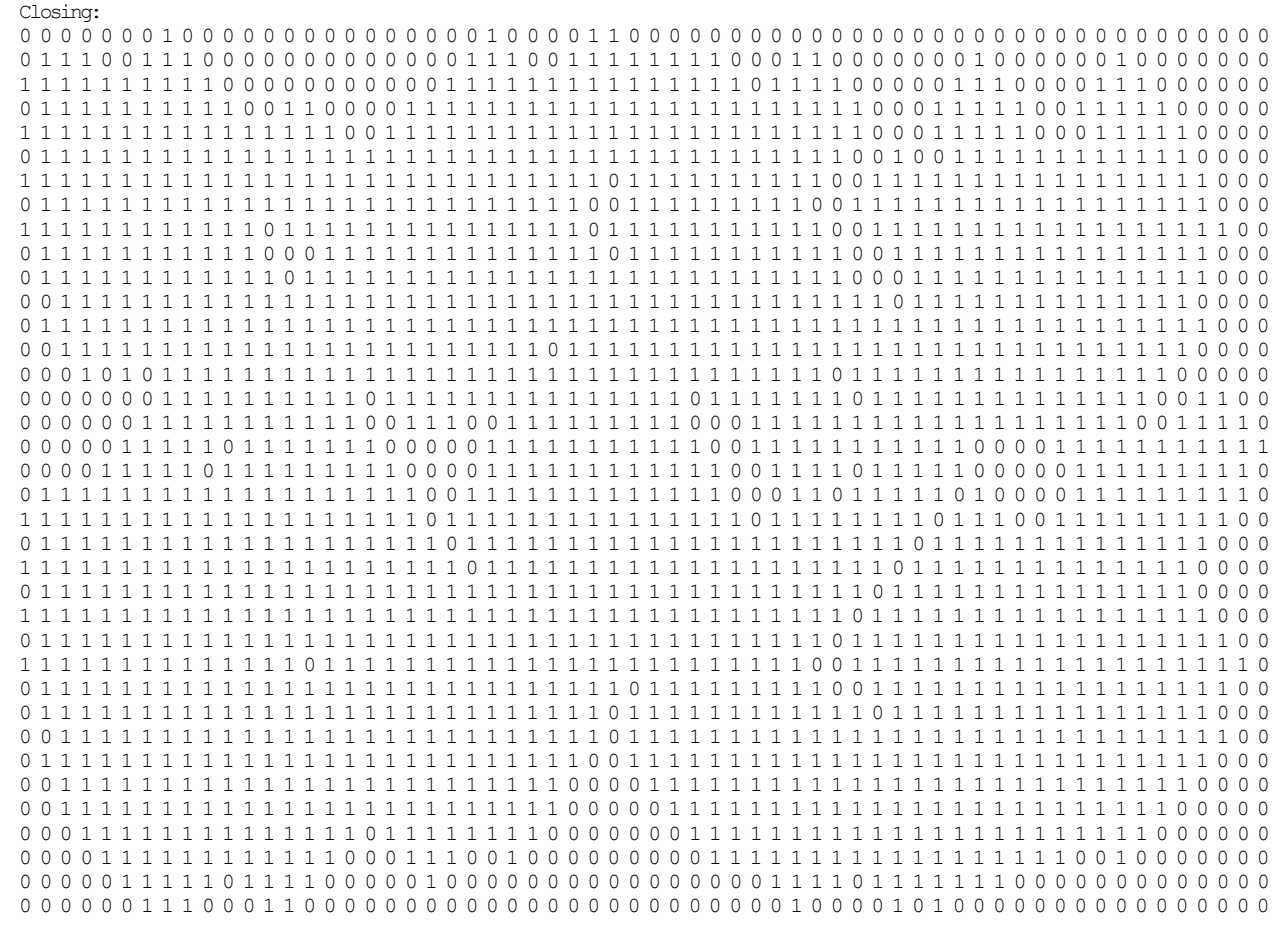
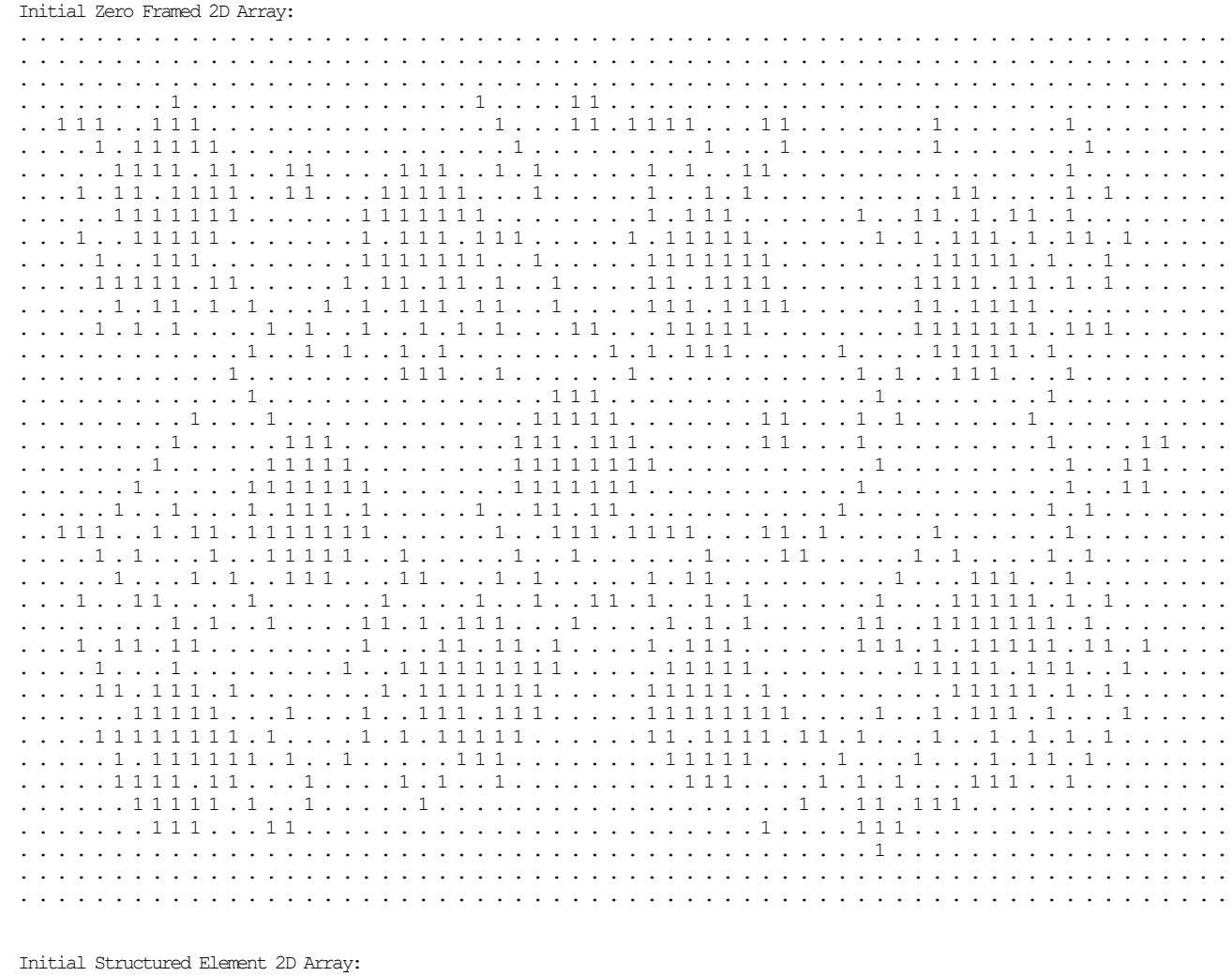


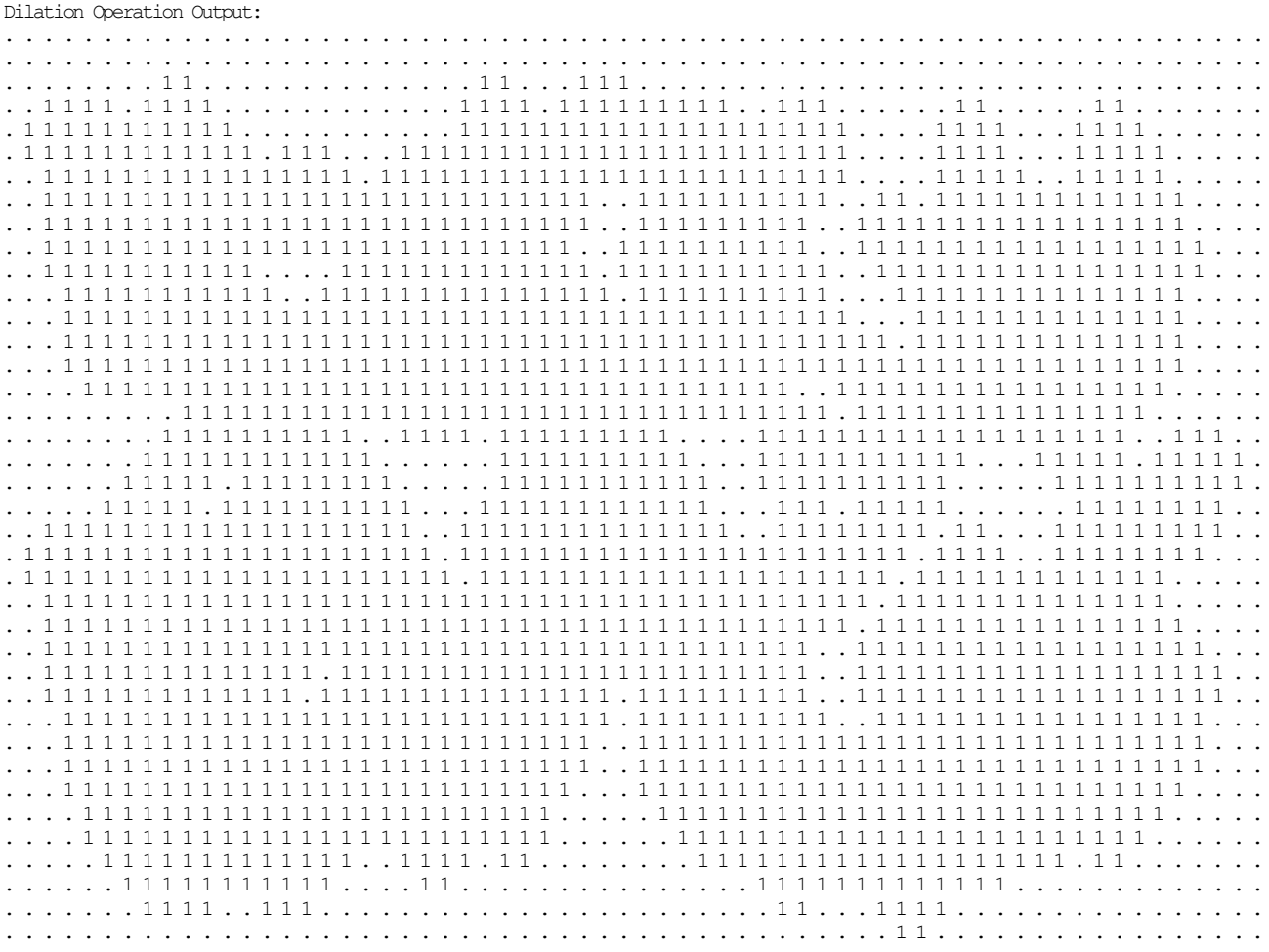
Image 1, Structured Element 3
****
Initial Zero Framed 2D Array:



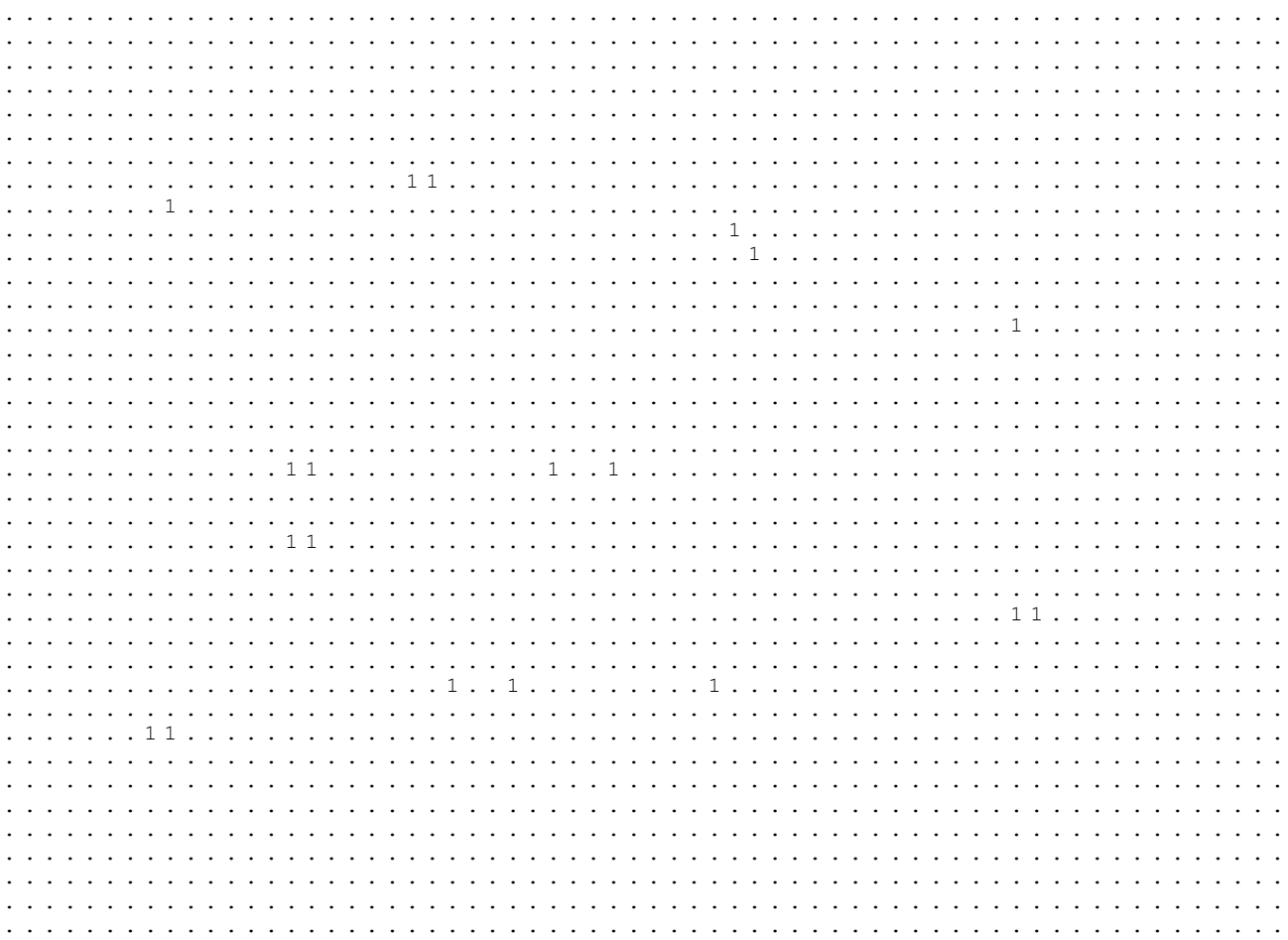
Initial Structured Element 2D Array:

```
. 1 1 .
1 1 1 1
1 1 1 1
. 1 1 .
```

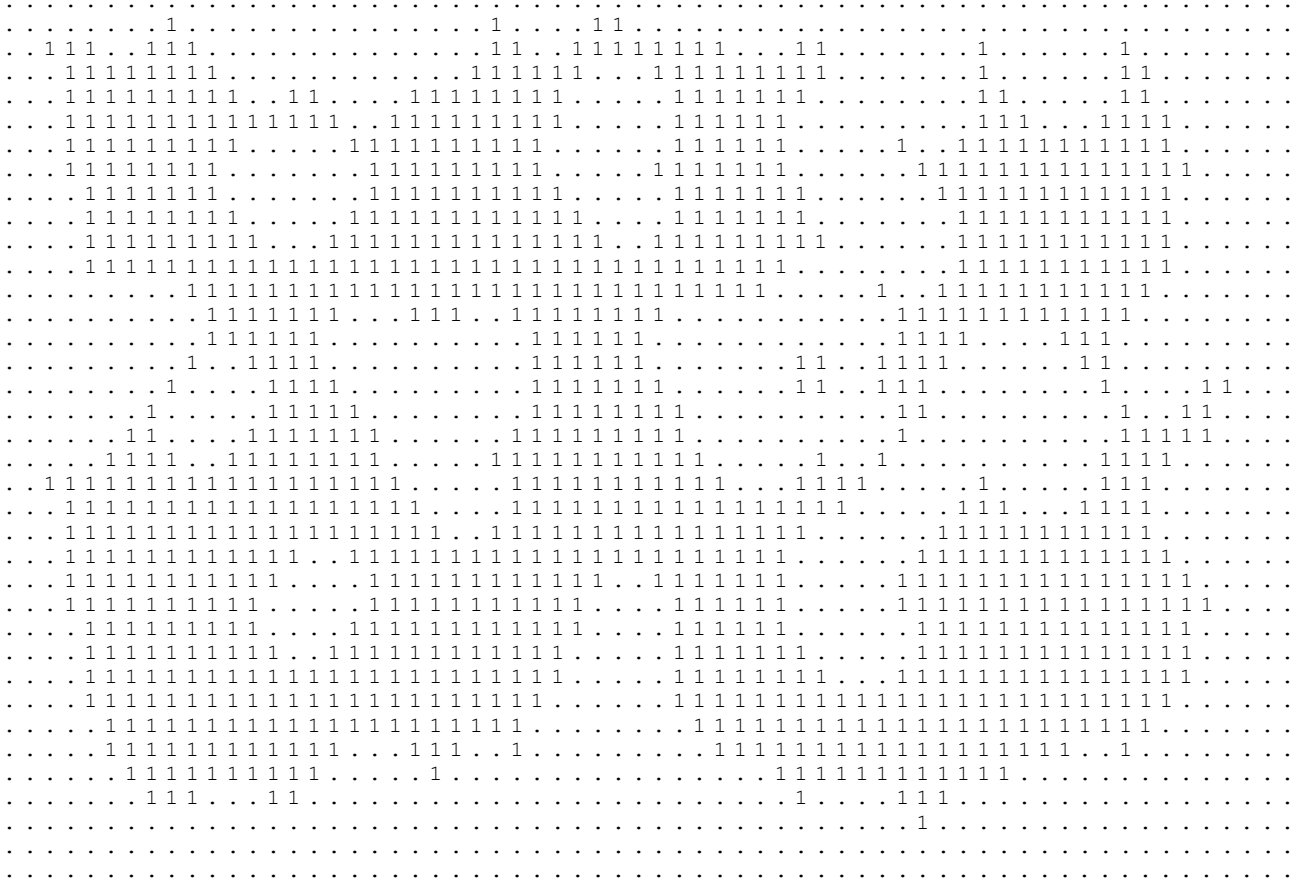
Dilation Operation Output:



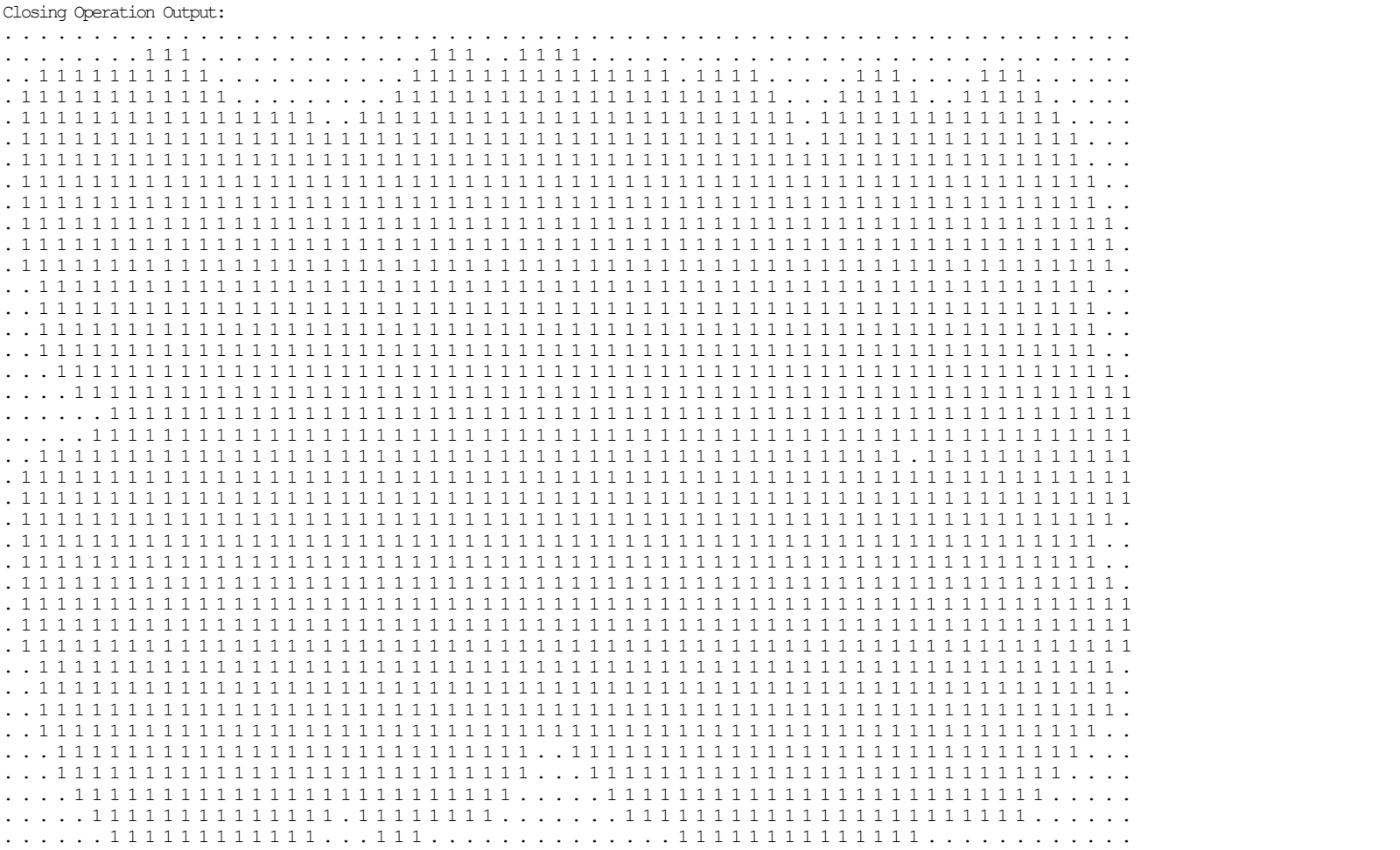
Erosion Operation Output:



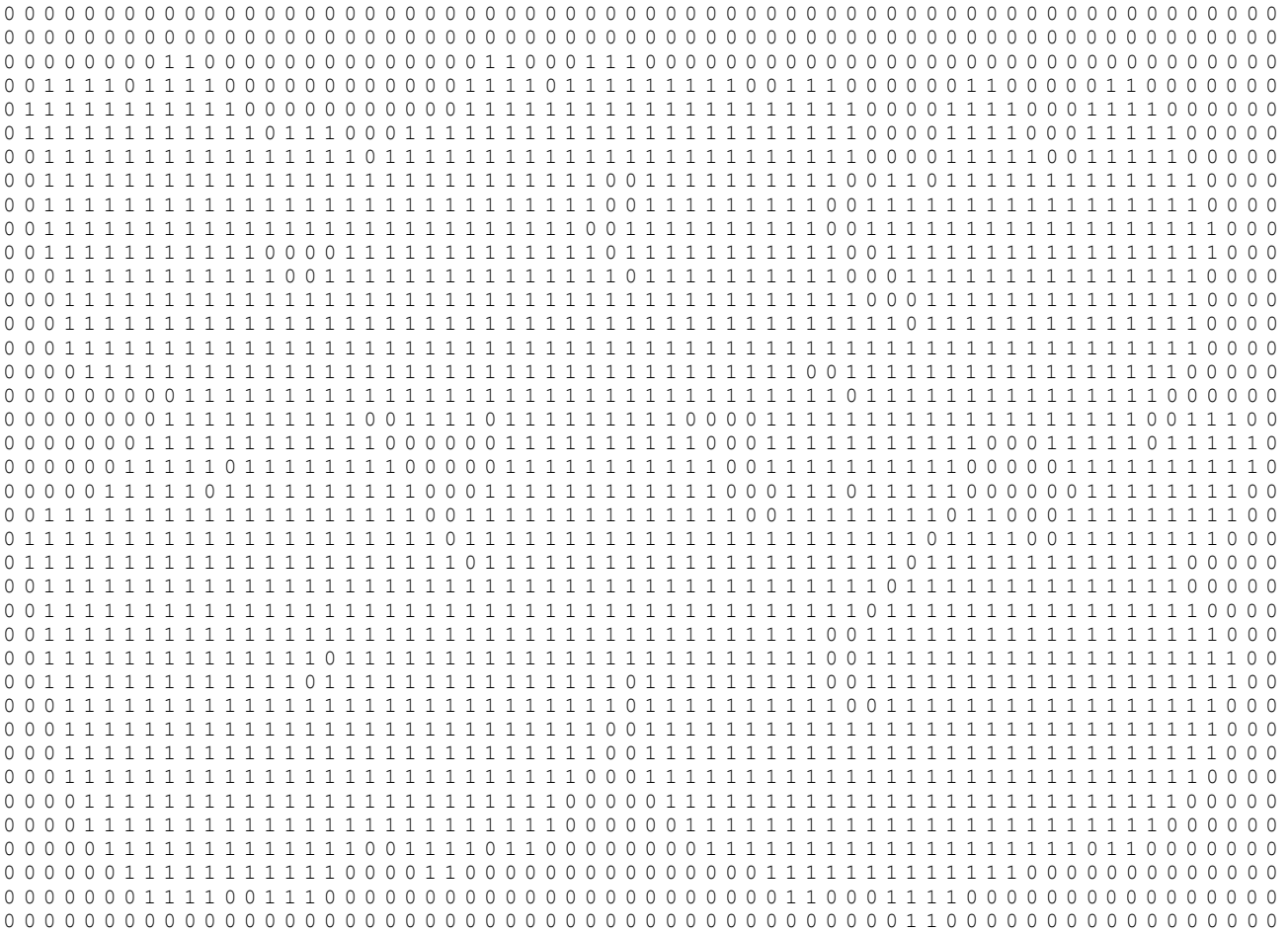
Opening Operation Output:



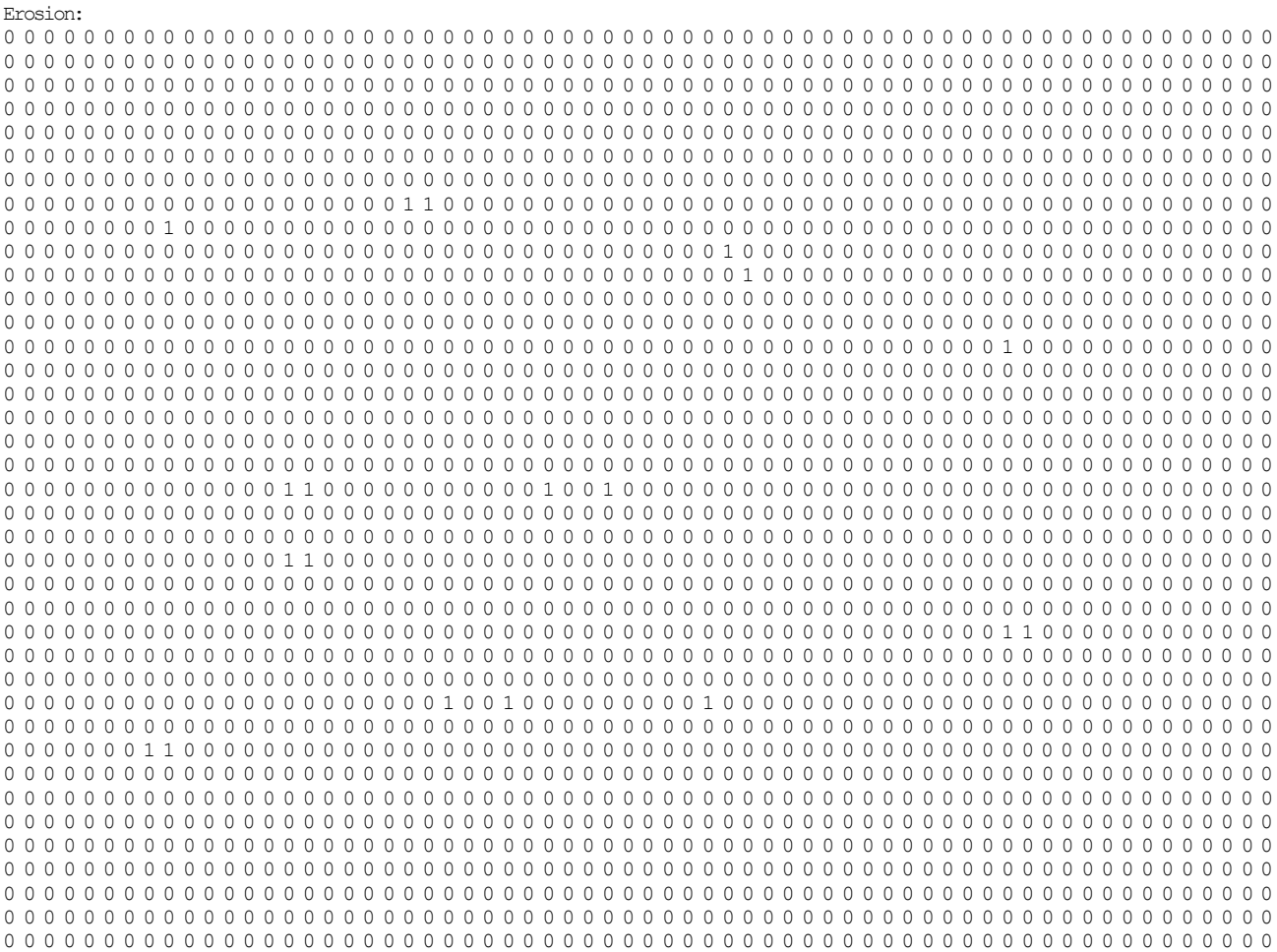
Closing Operation Output:



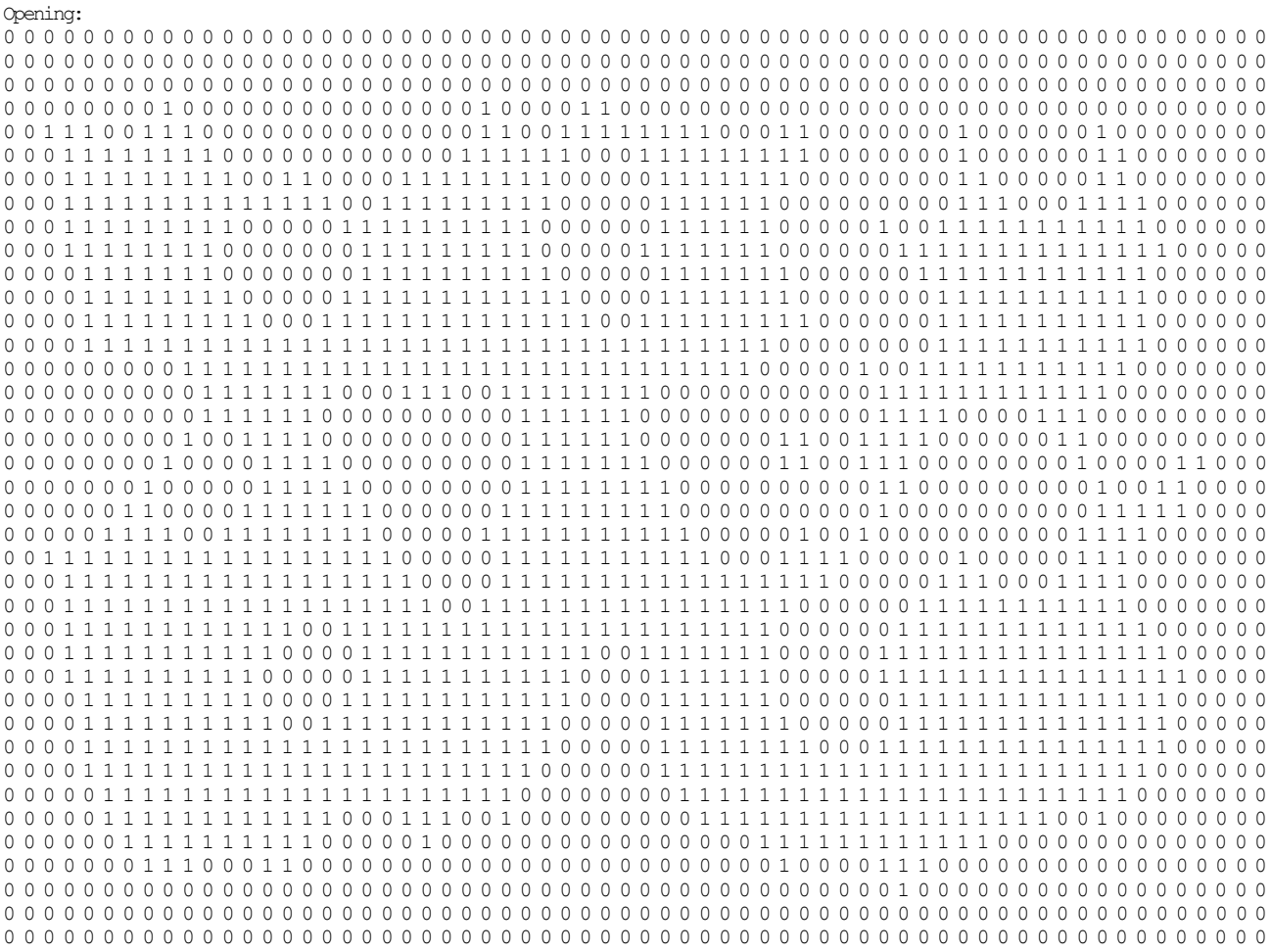
Dilation:



Erosion:



Opening:



Closing:

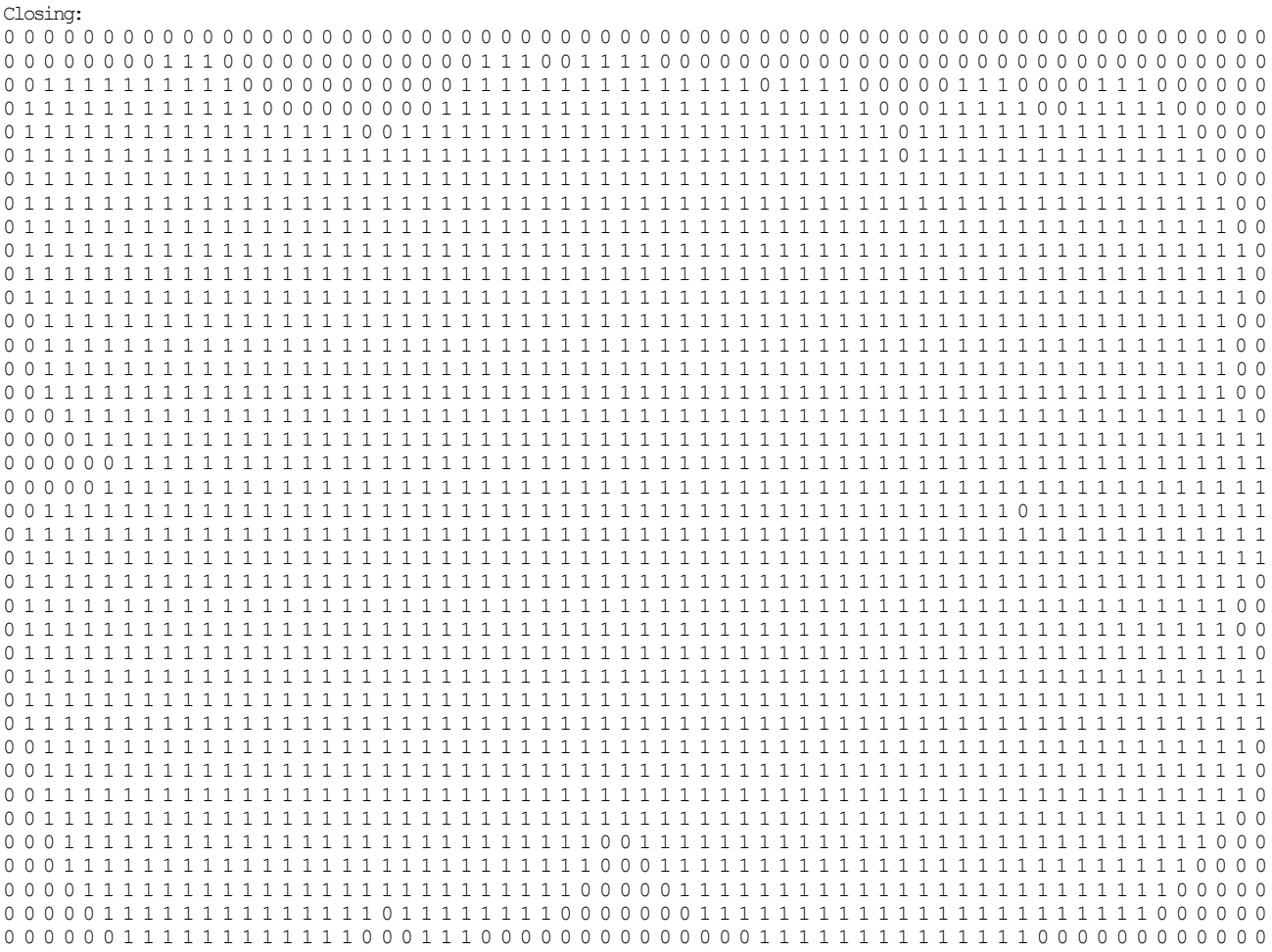
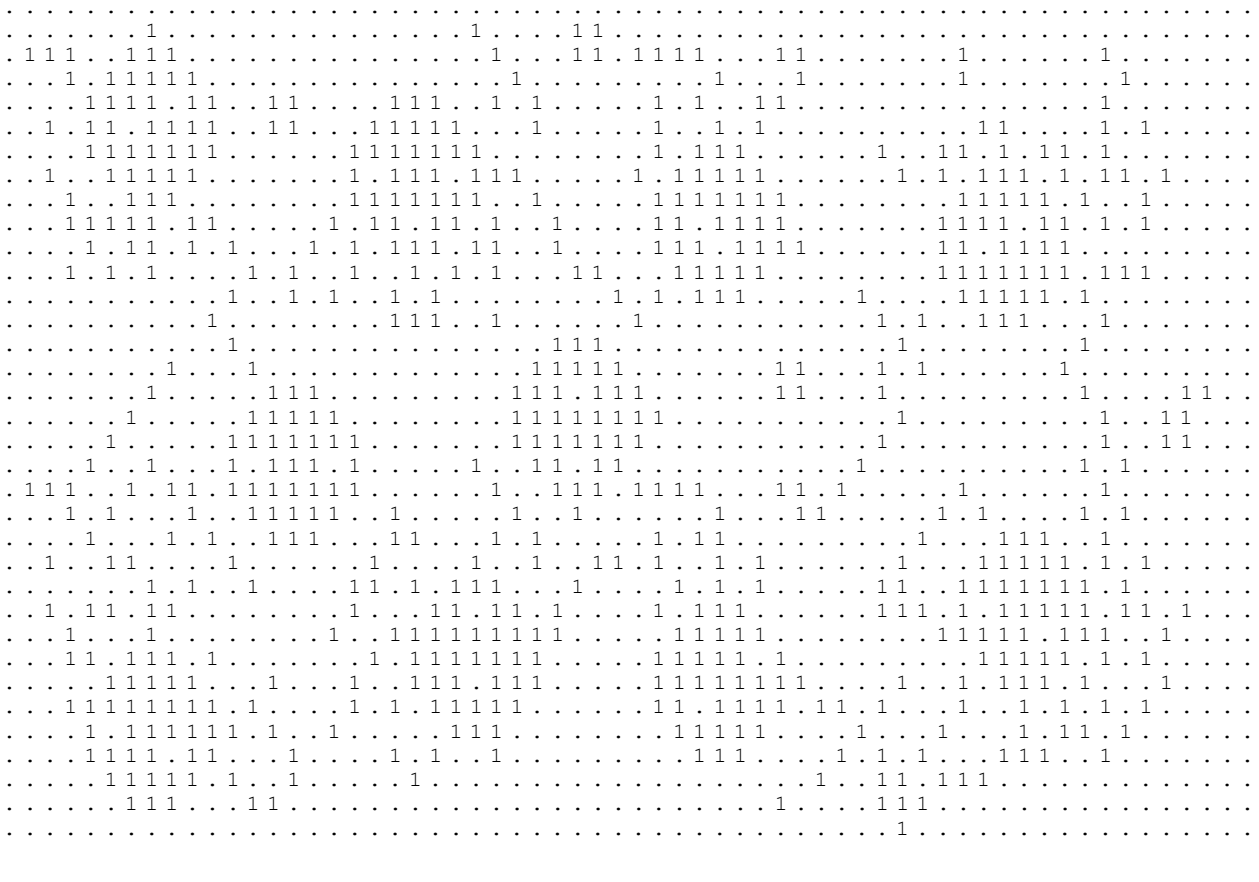
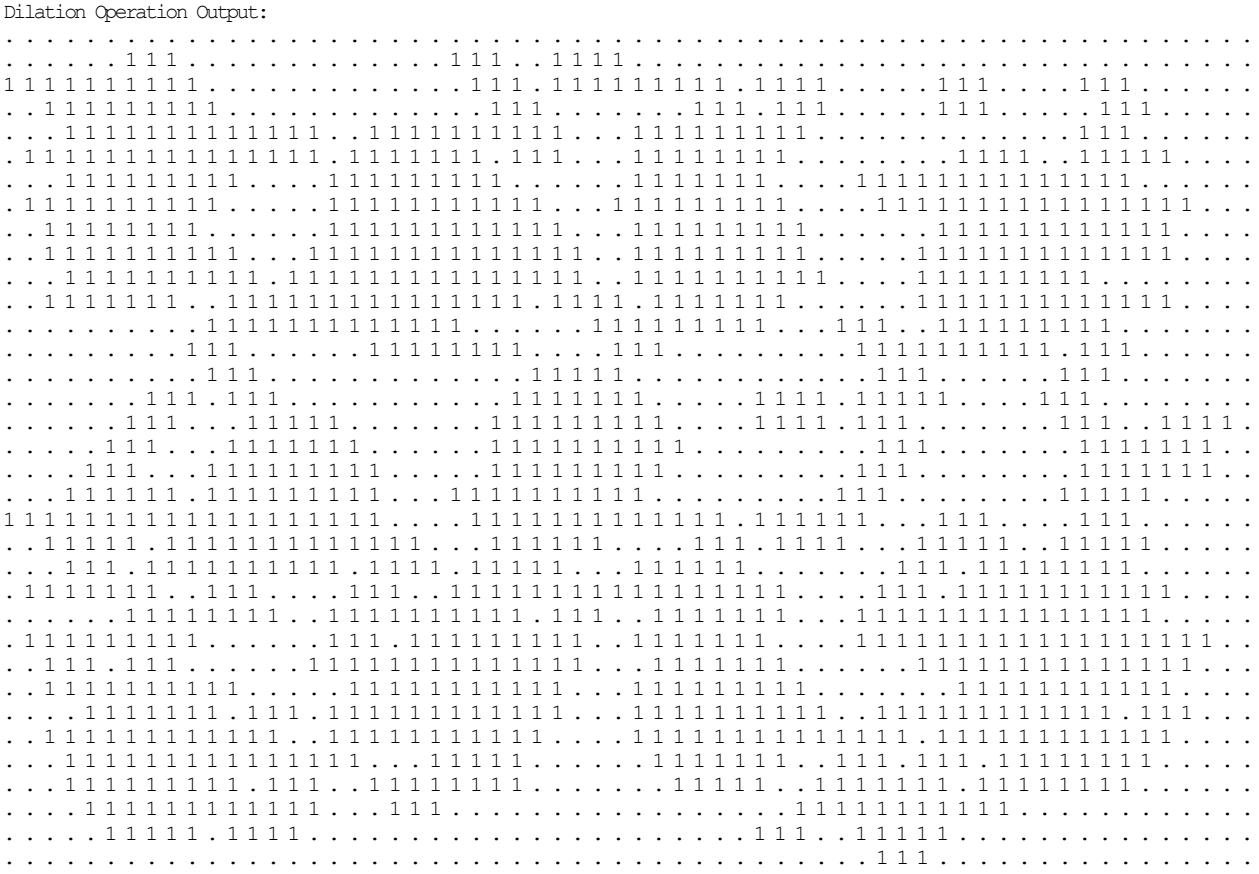


Image 2, Structured Element 1
****
Initial Zero Framed 2D Array:

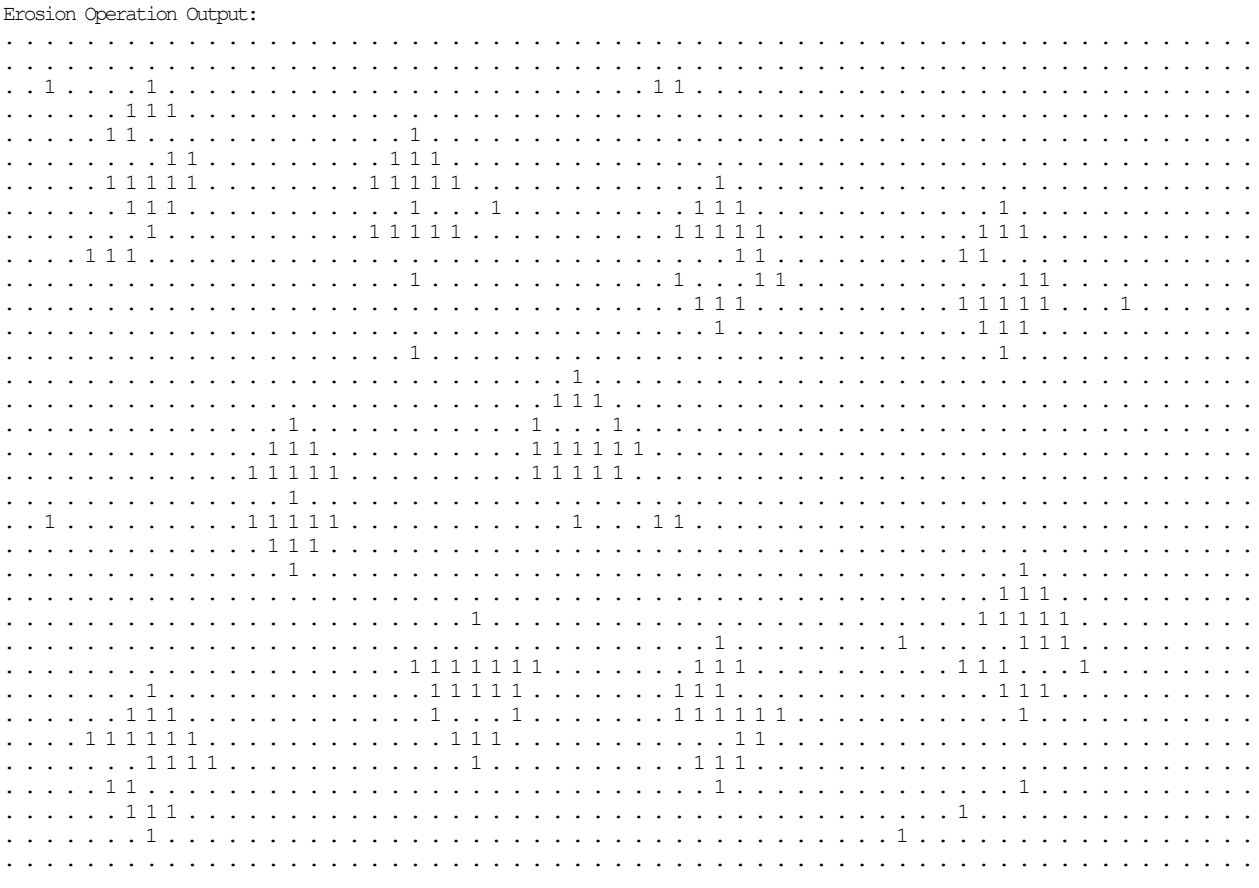


Initial Structured Element 2D Array:
1 1 1

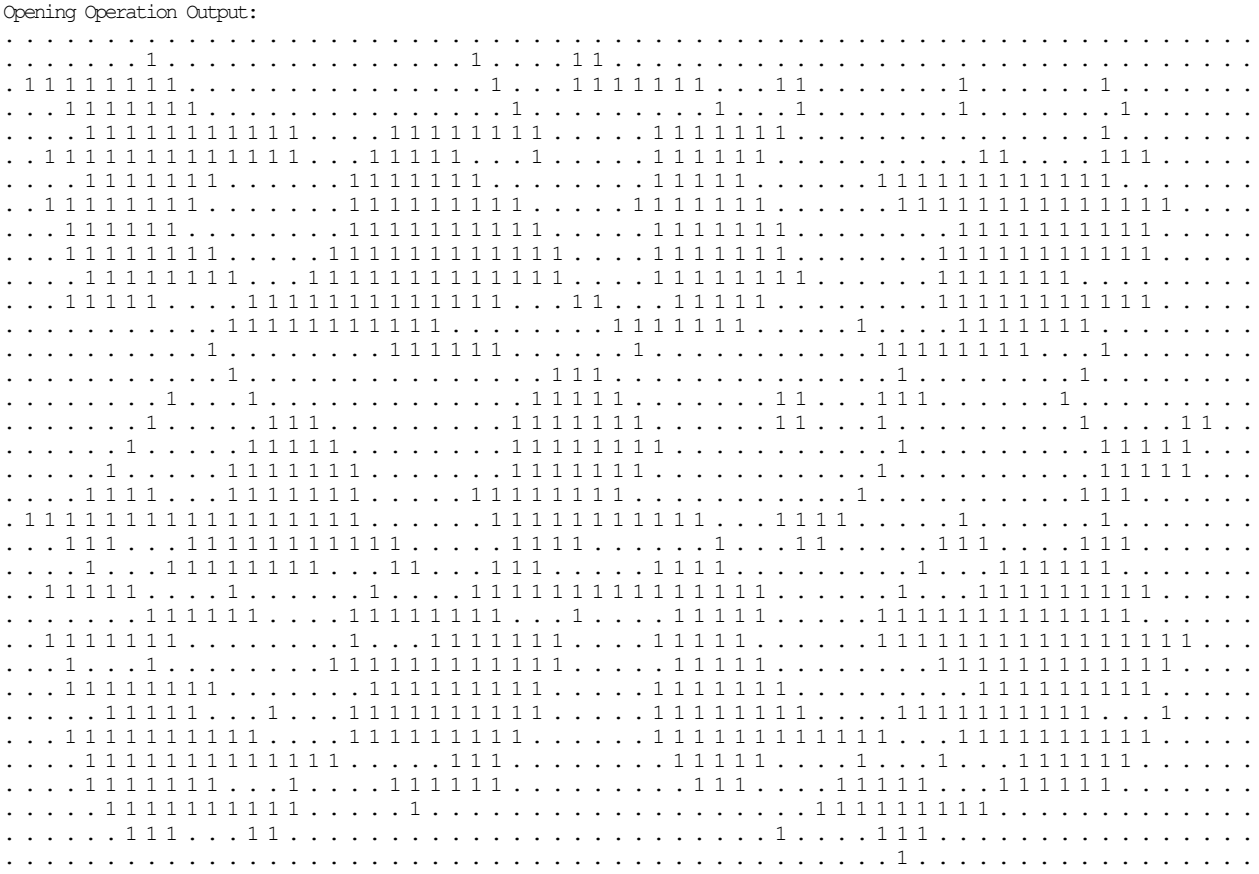
Dilation Operation Output:



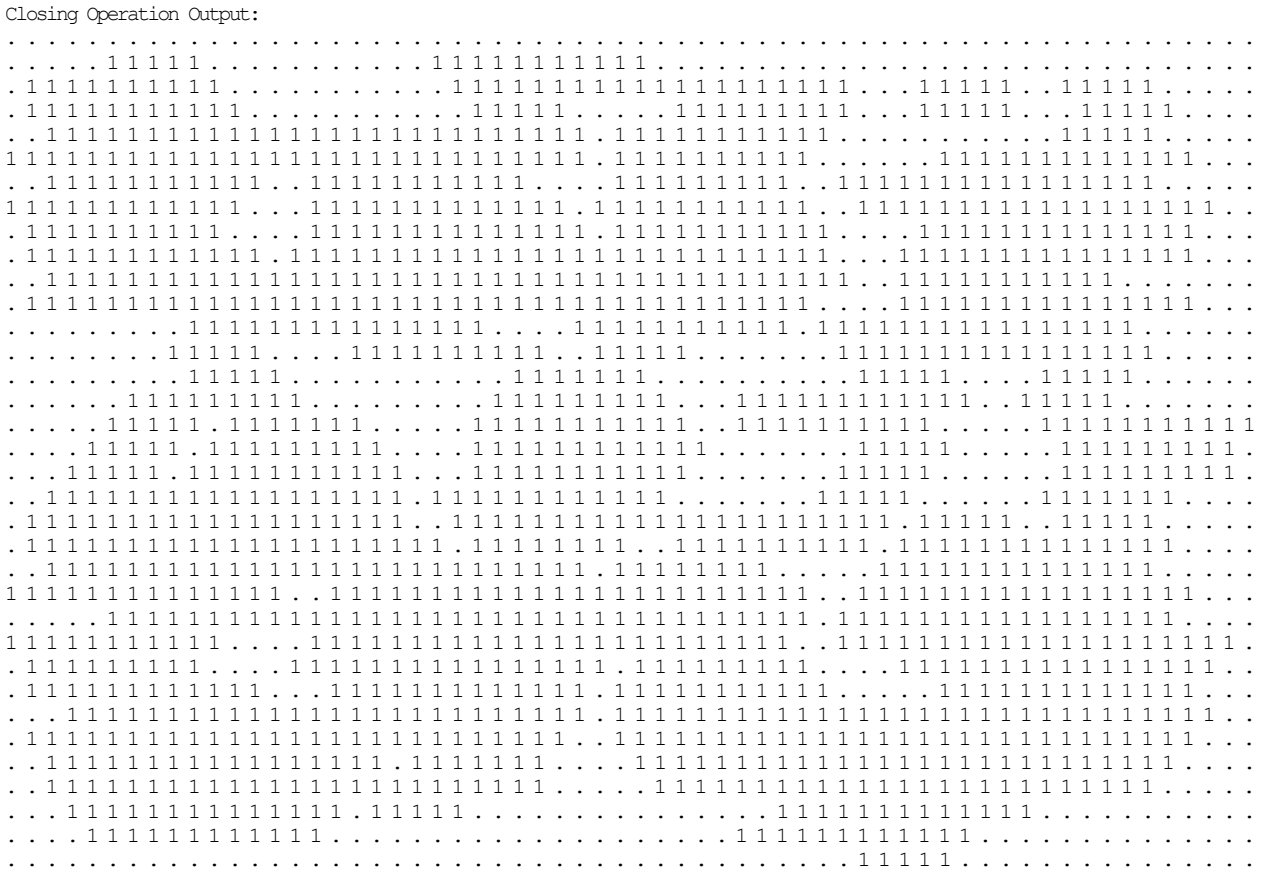
Erosion Operation Output:



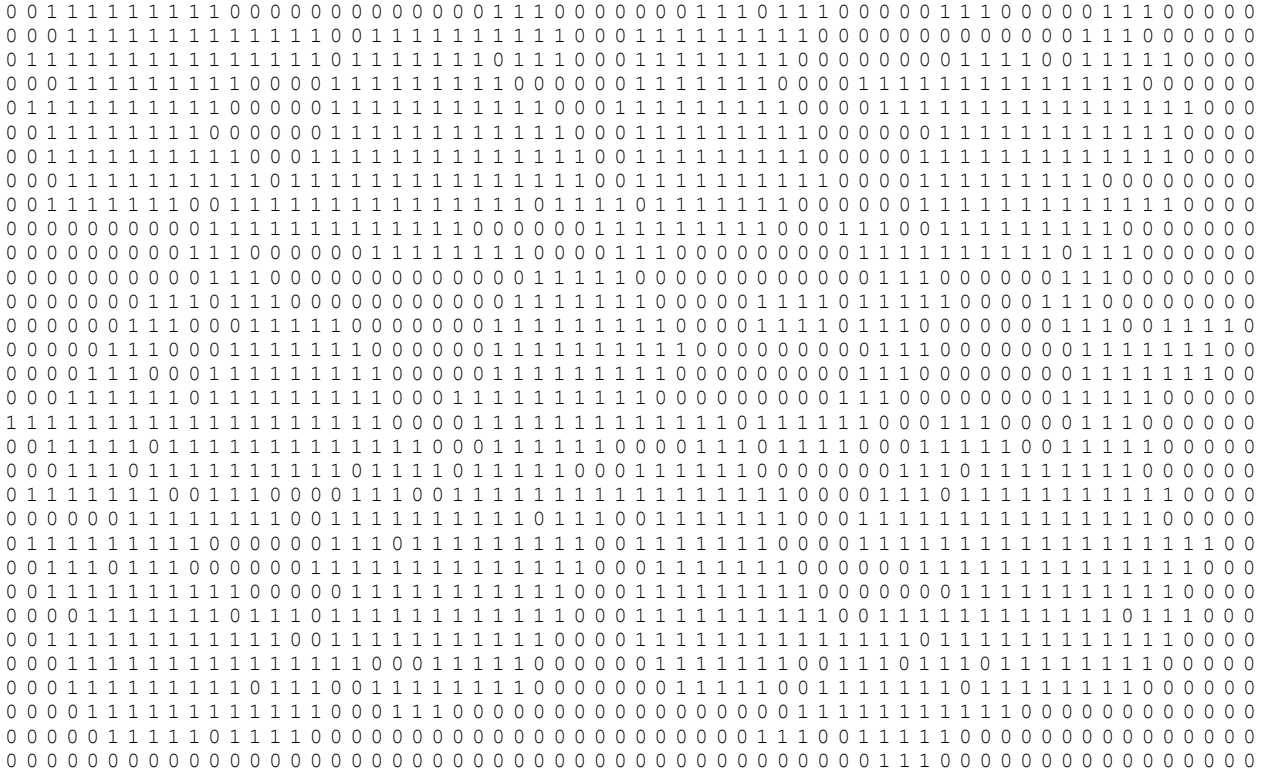
Opening Operation Output:



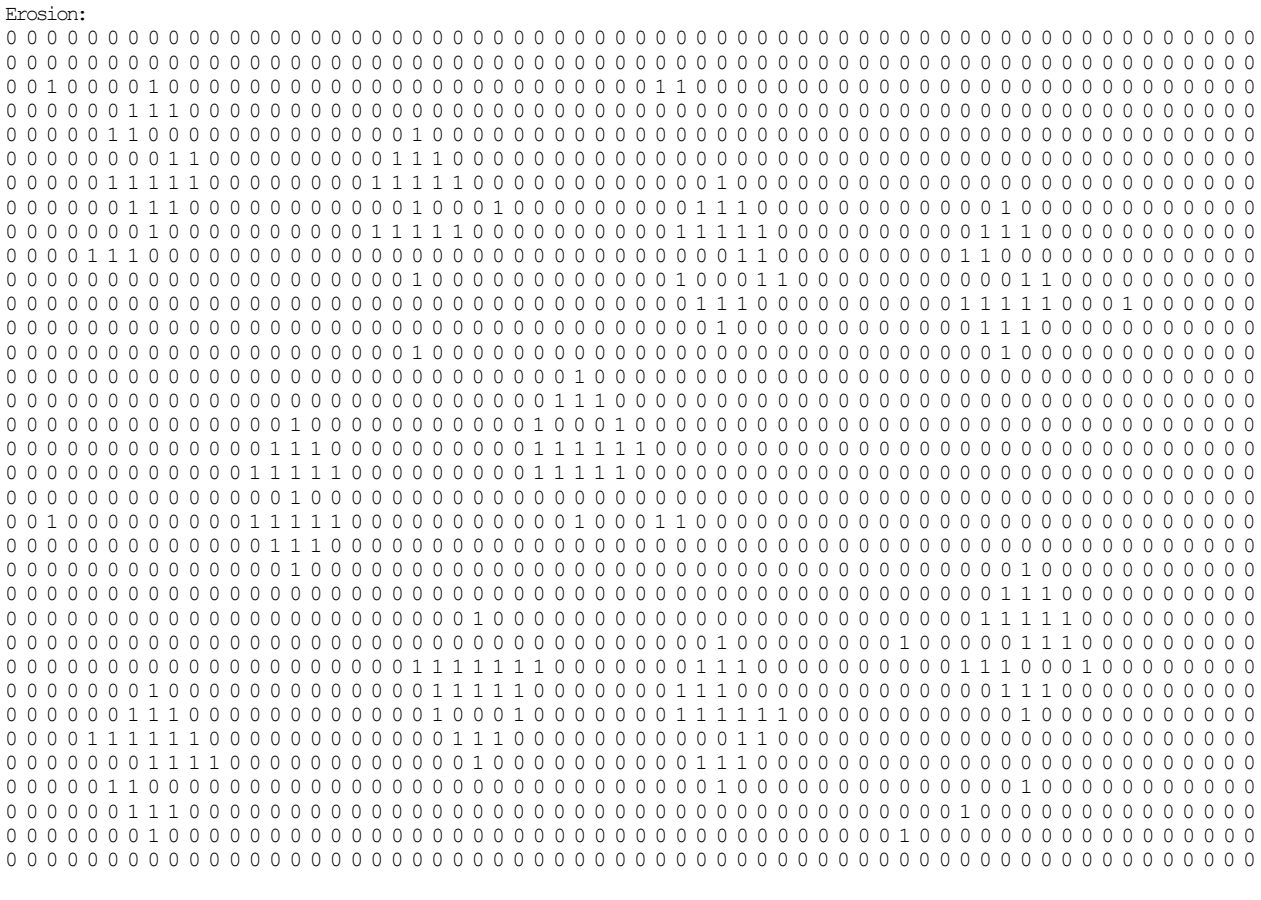
Closing Operation Output:



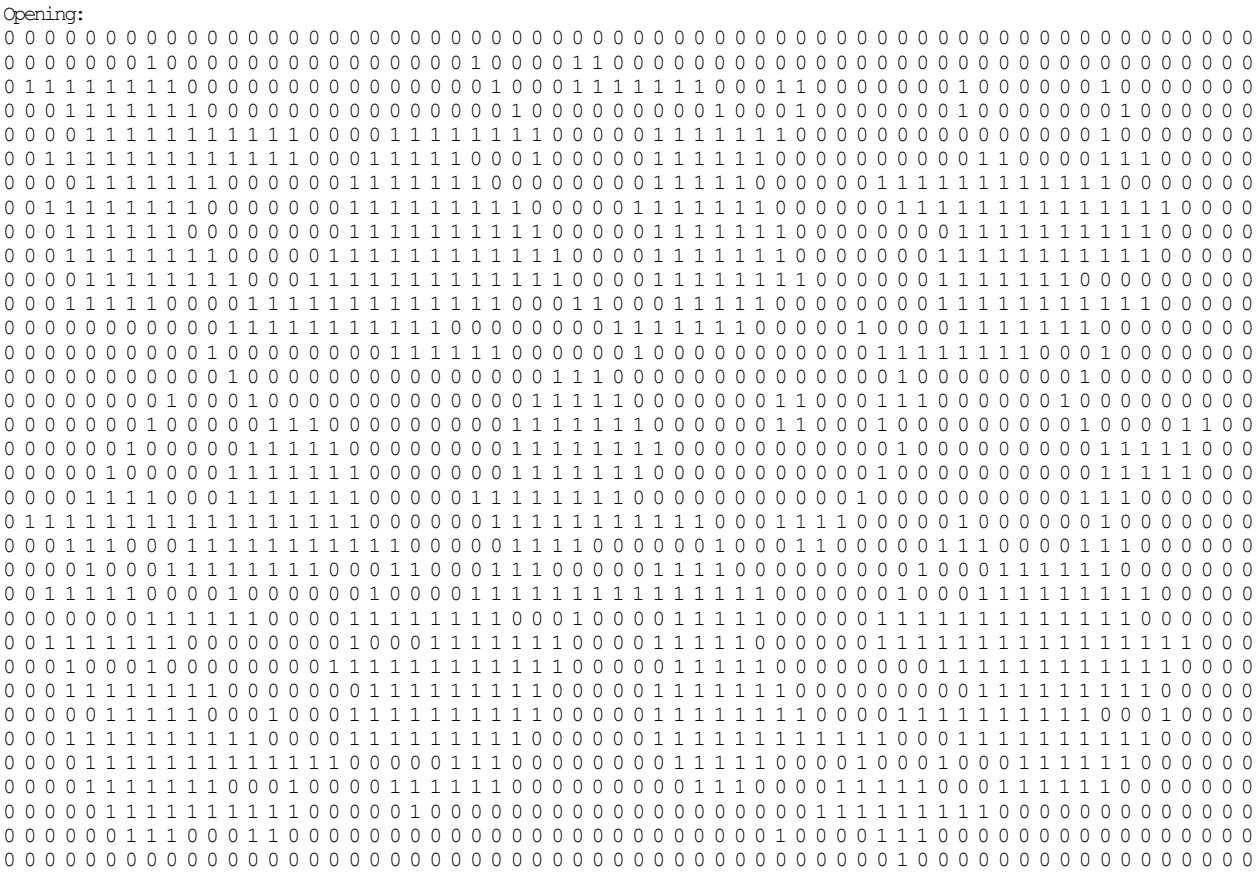
Dilation:



Erosion:



Opening:



Closing:

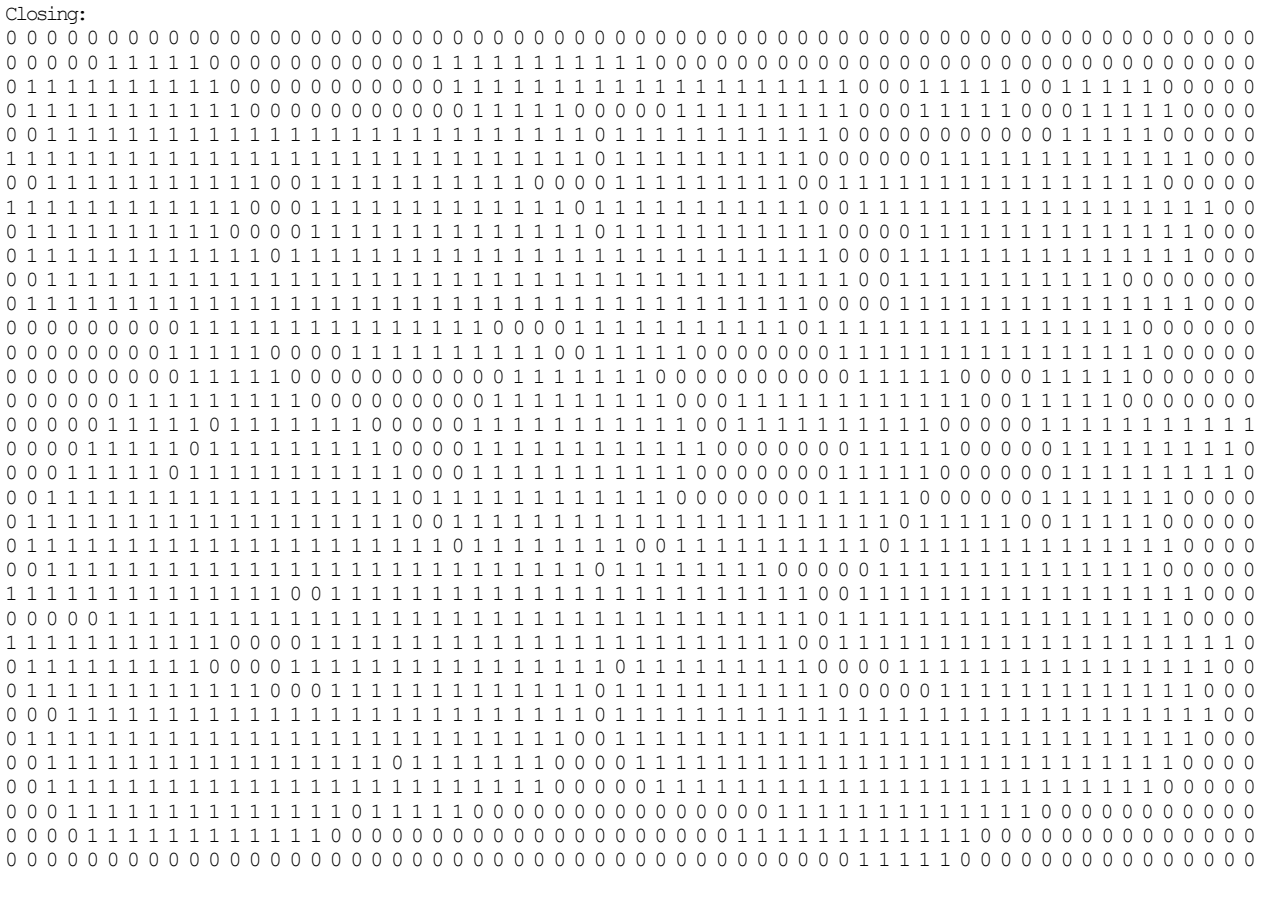
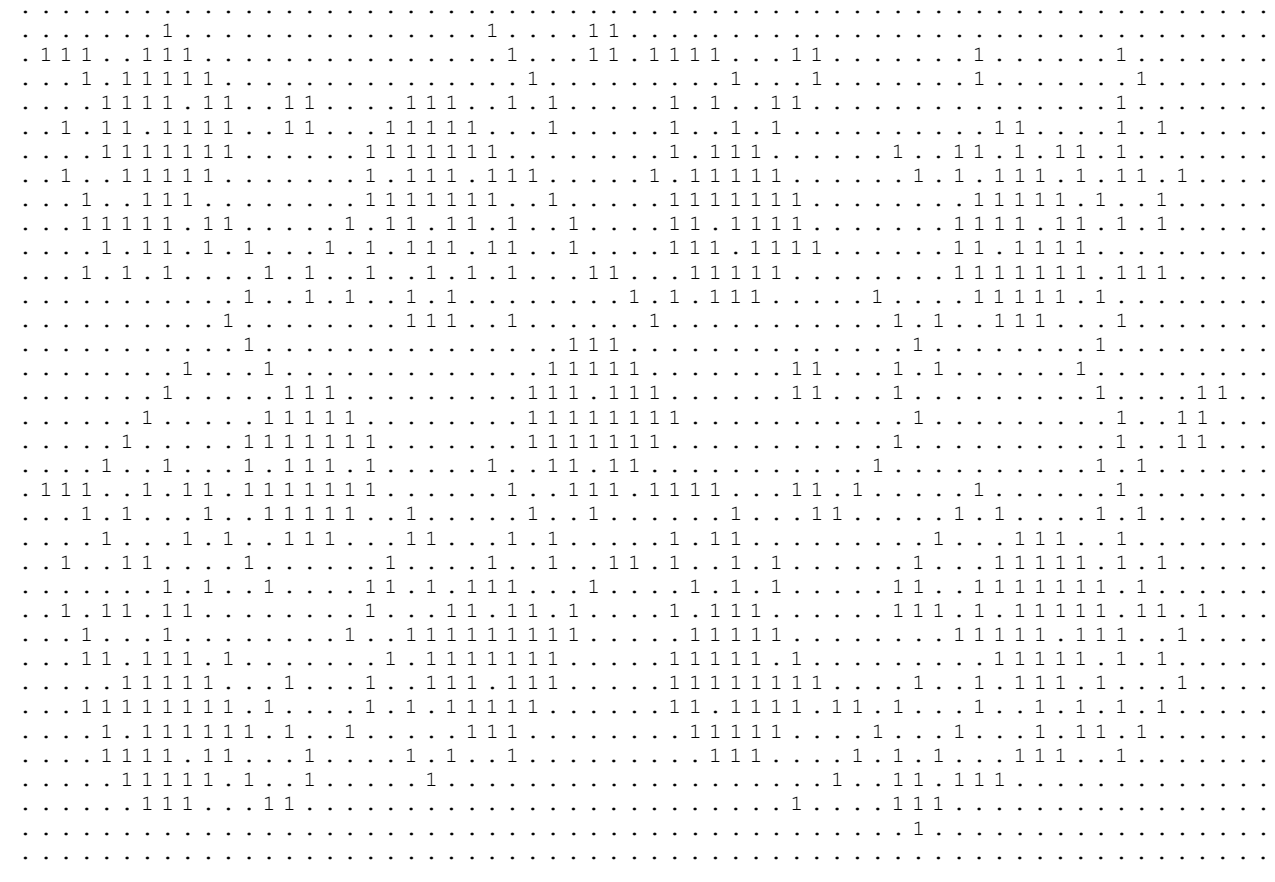


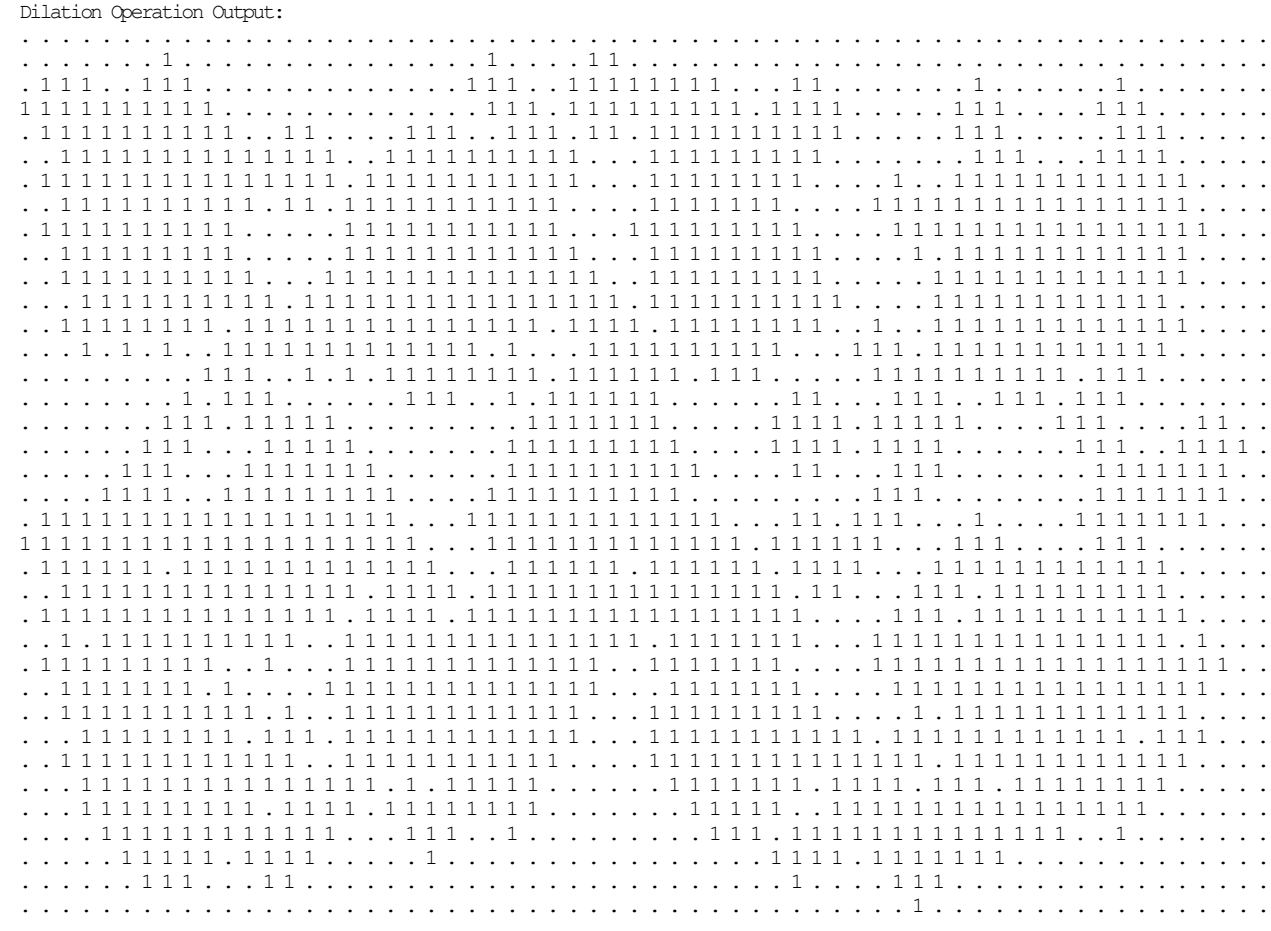
Image 2, Structured Element 2
****
Initial Zero Framed 2D Array:



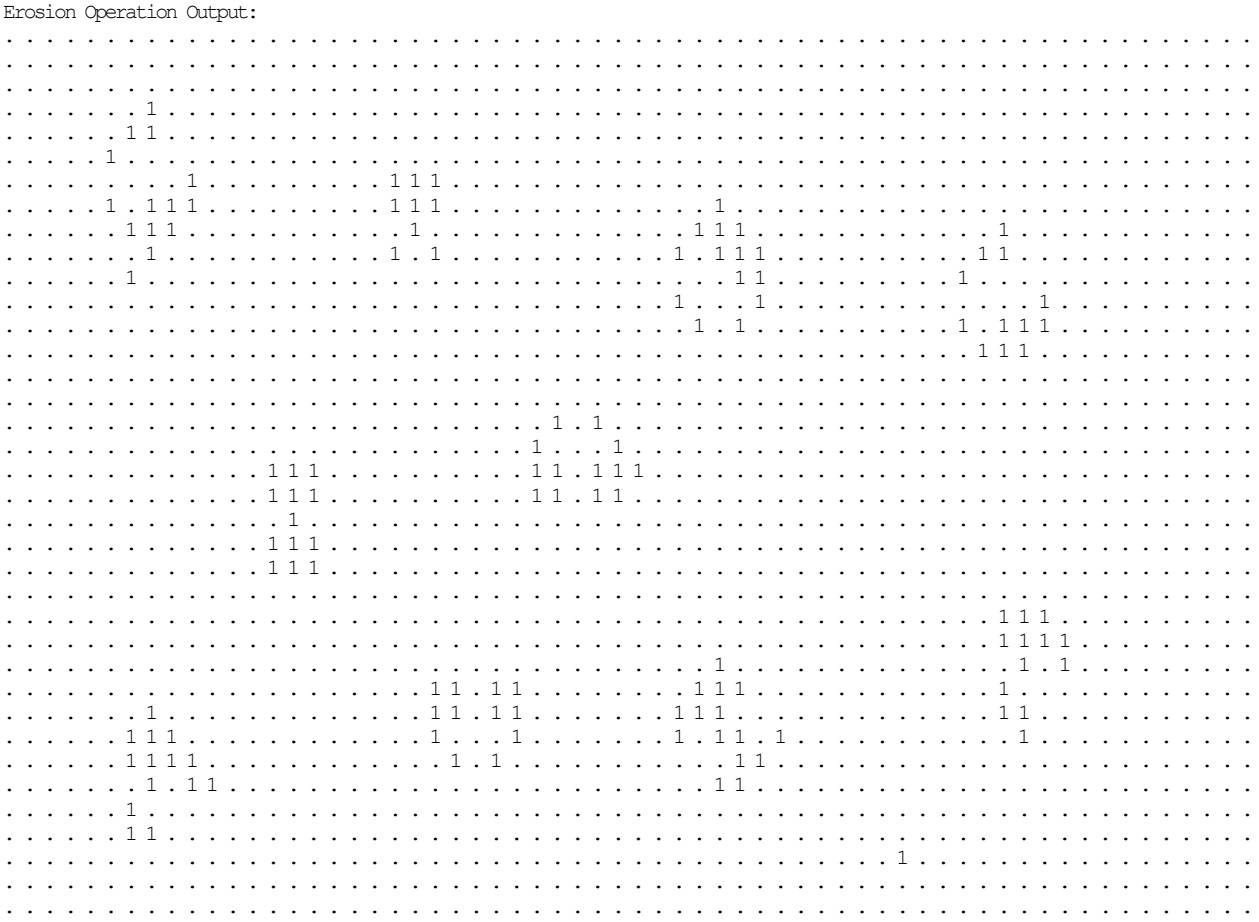
Initial Structured Element 2D Array:

```
. 1 .
1 1 1
. 1 .
```

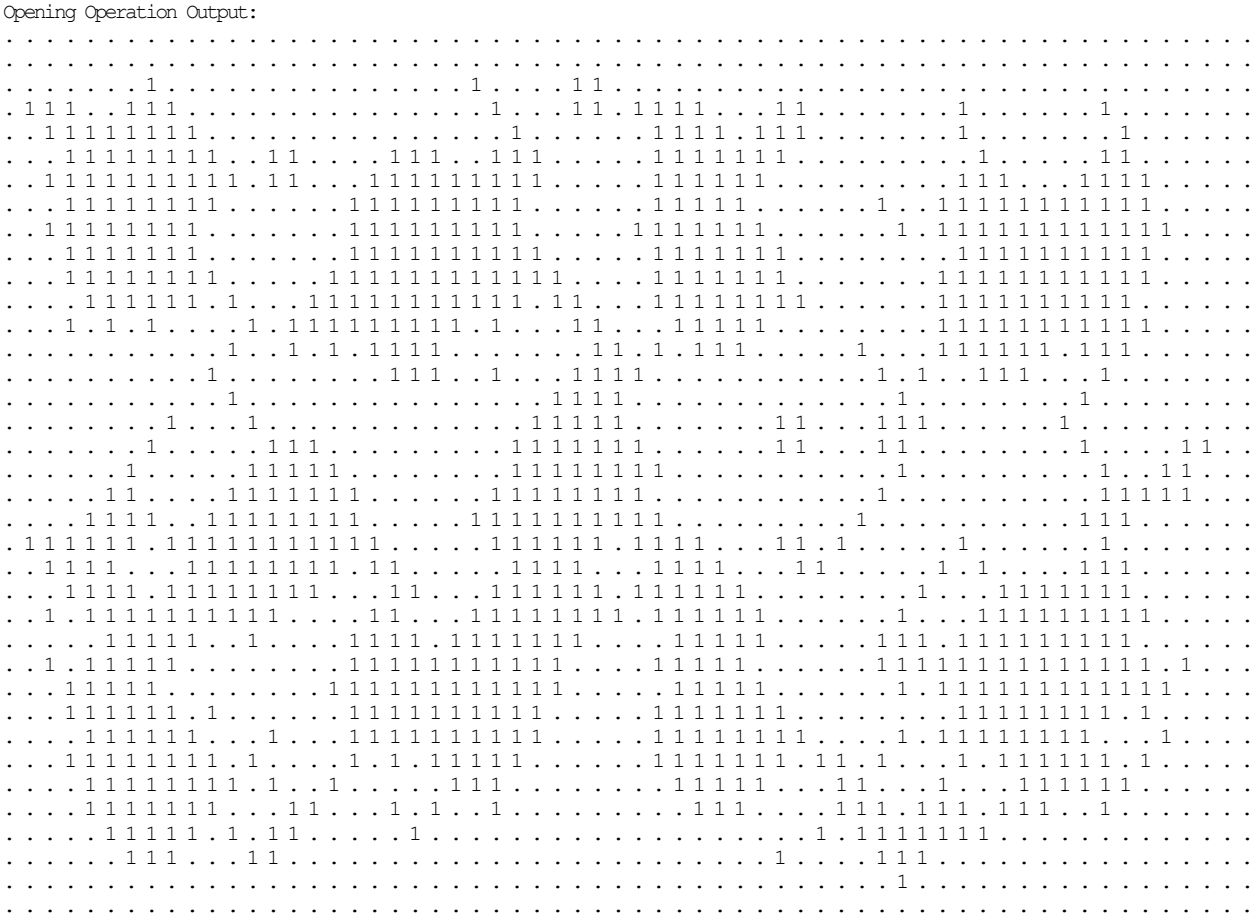
Dilation Operation Output:



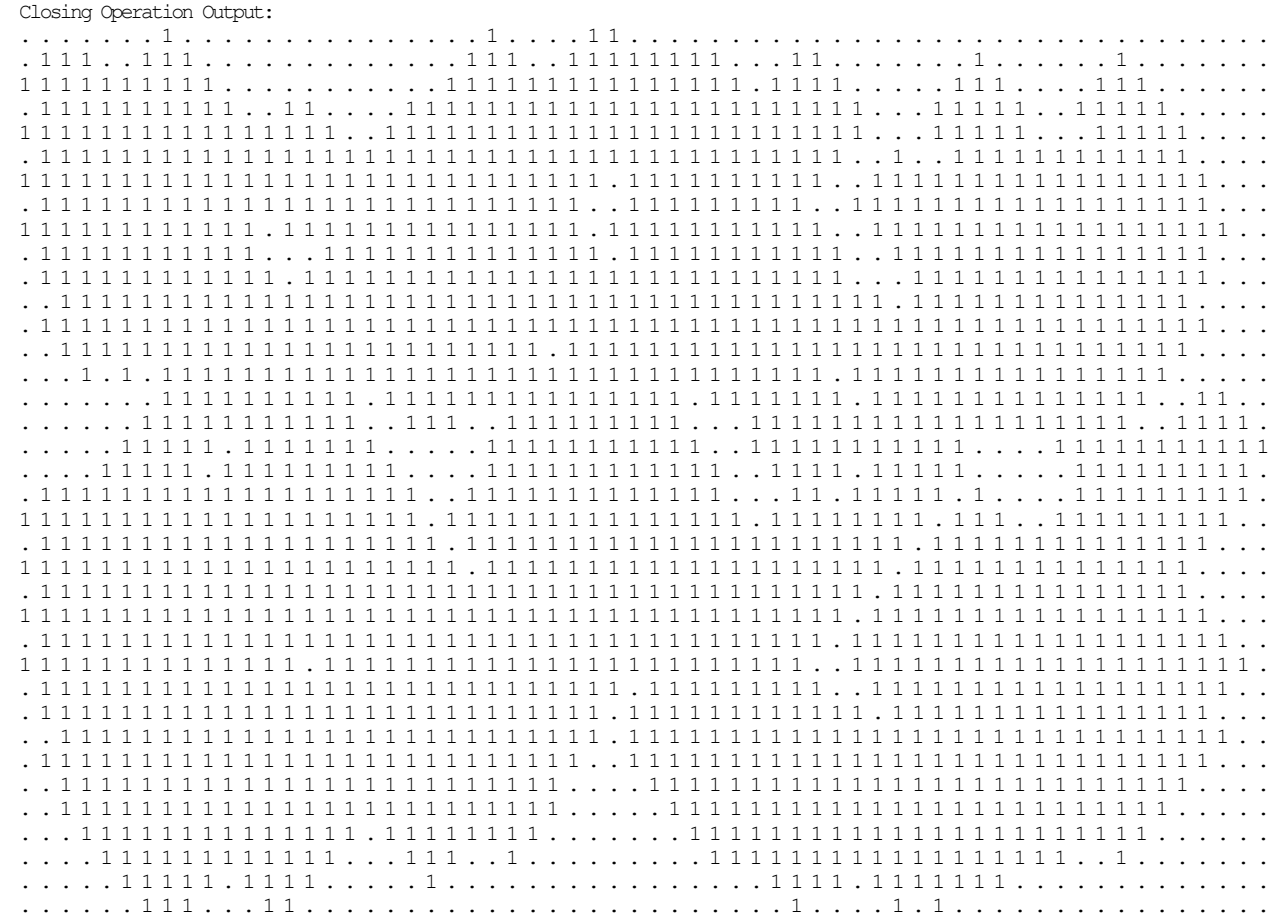
Erosion Operation Output:



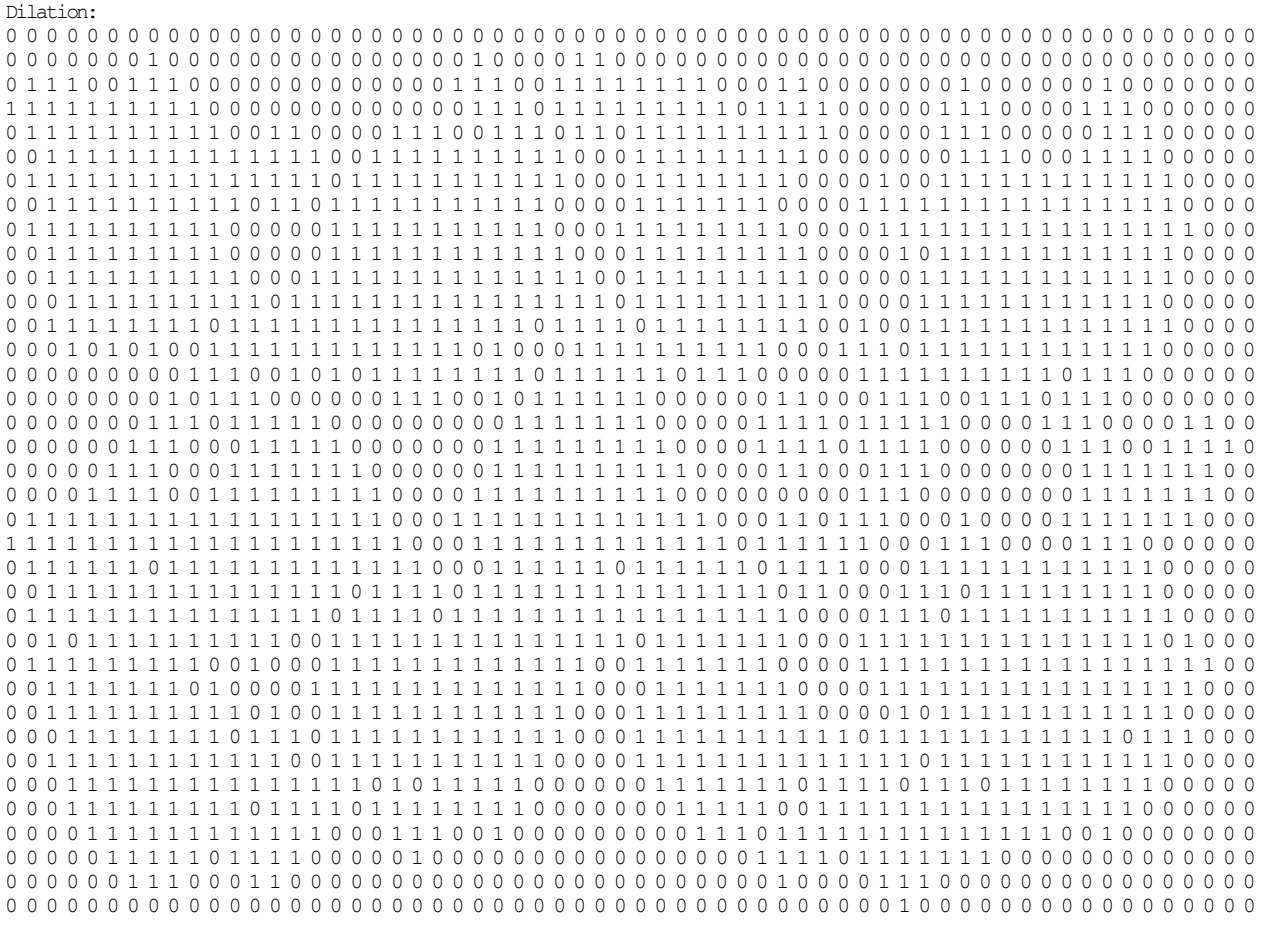
Opening Operation Output:



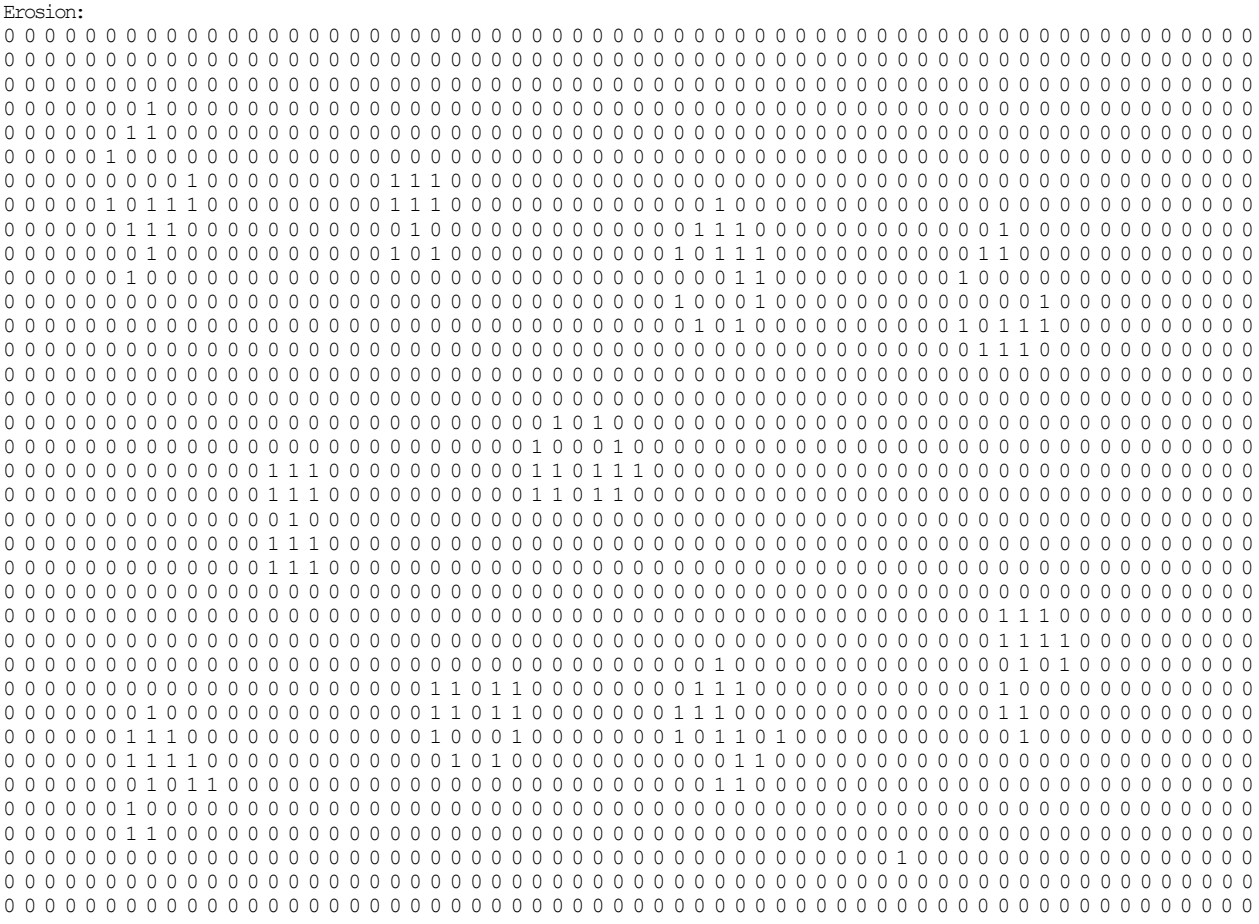
Closing Operation Output:



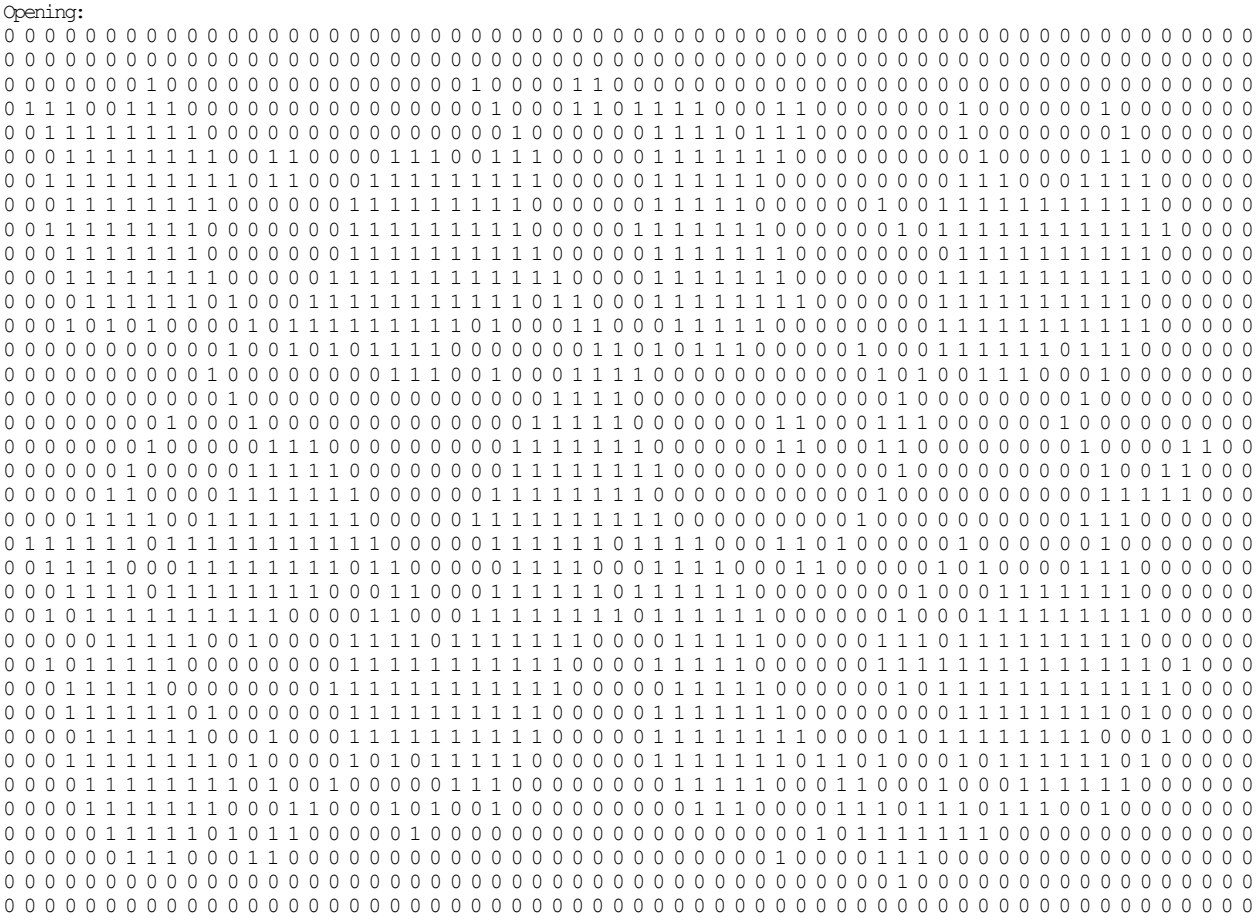
Dilation:



Erosion:



Opening:



Closing:

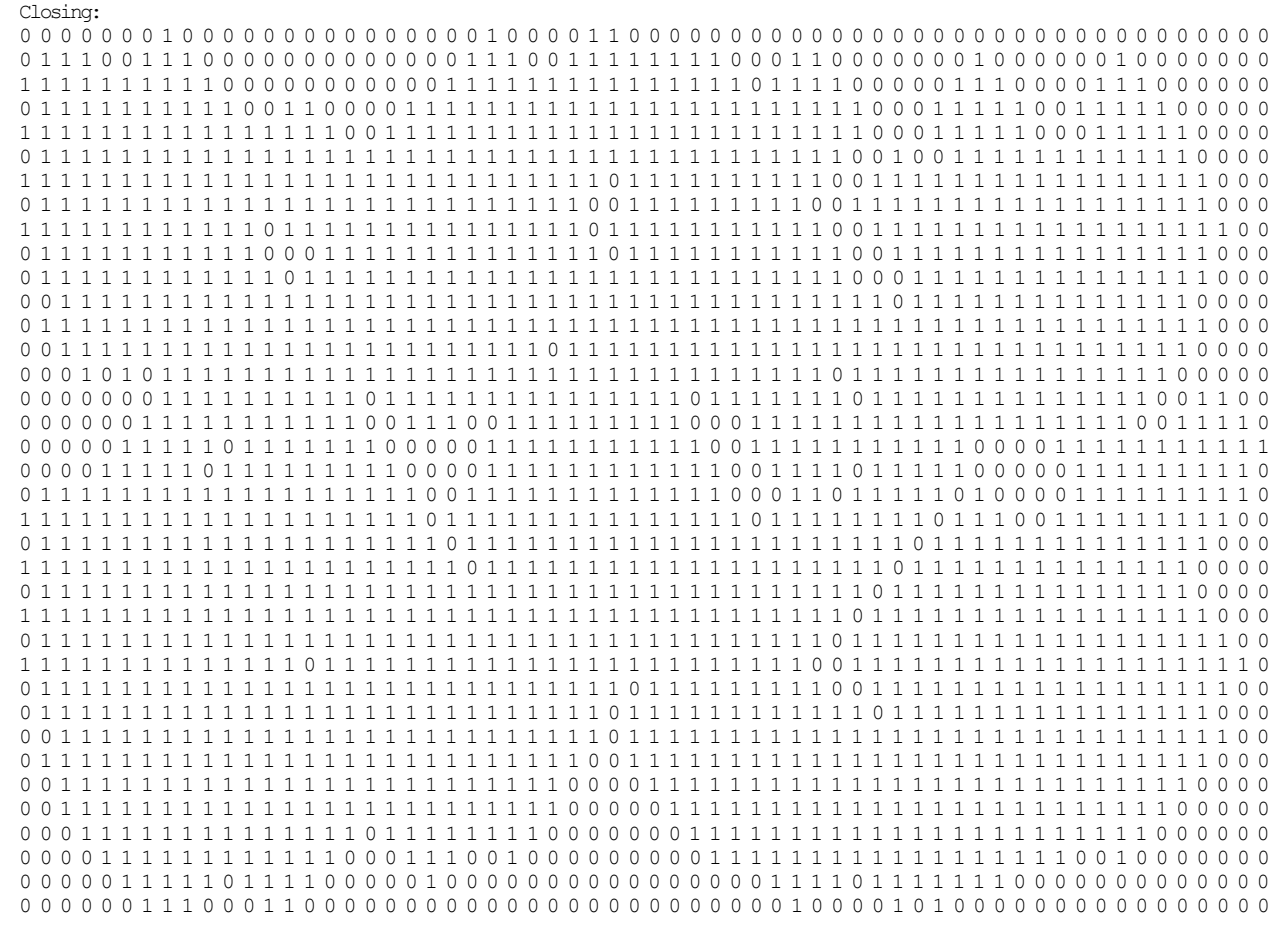
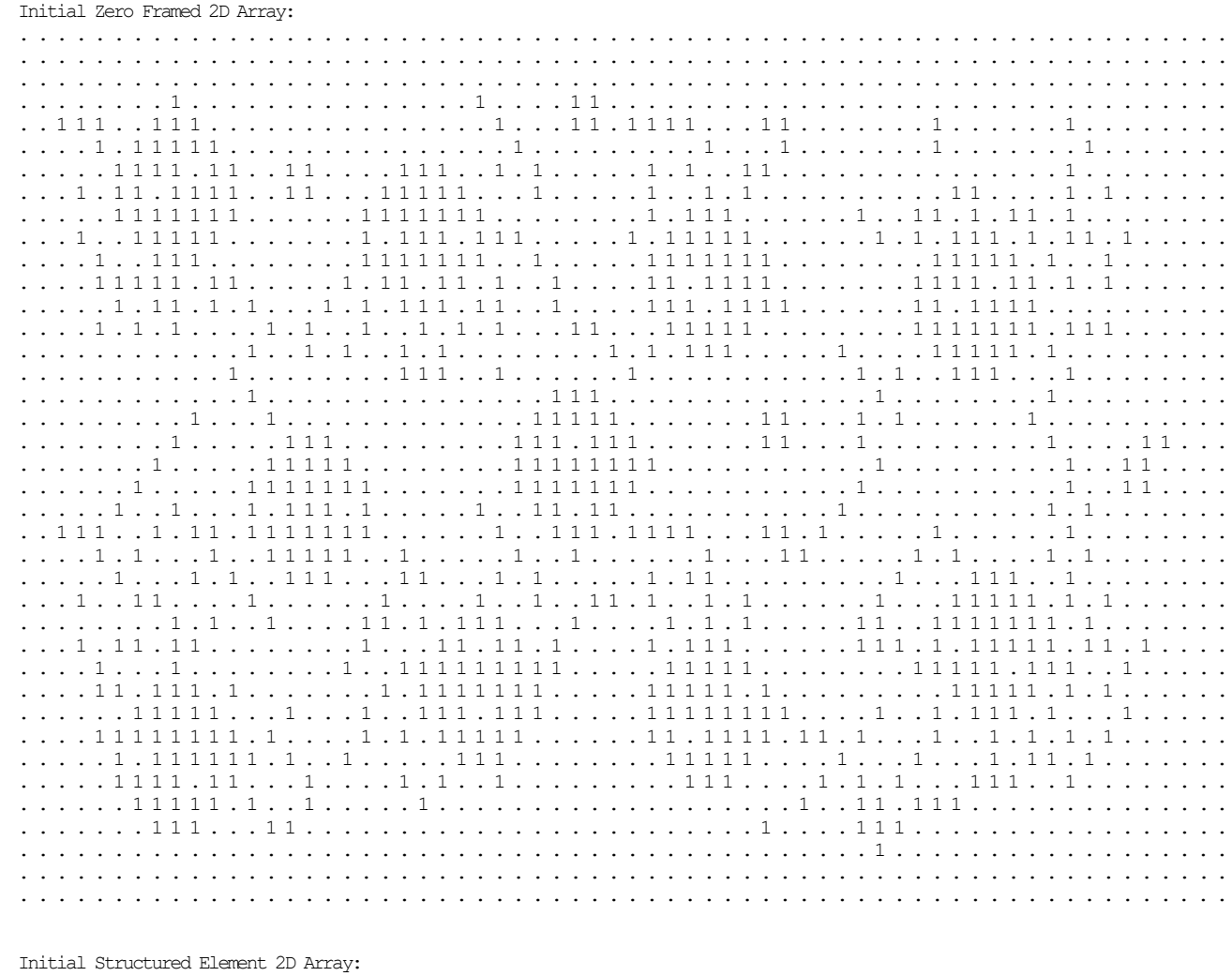


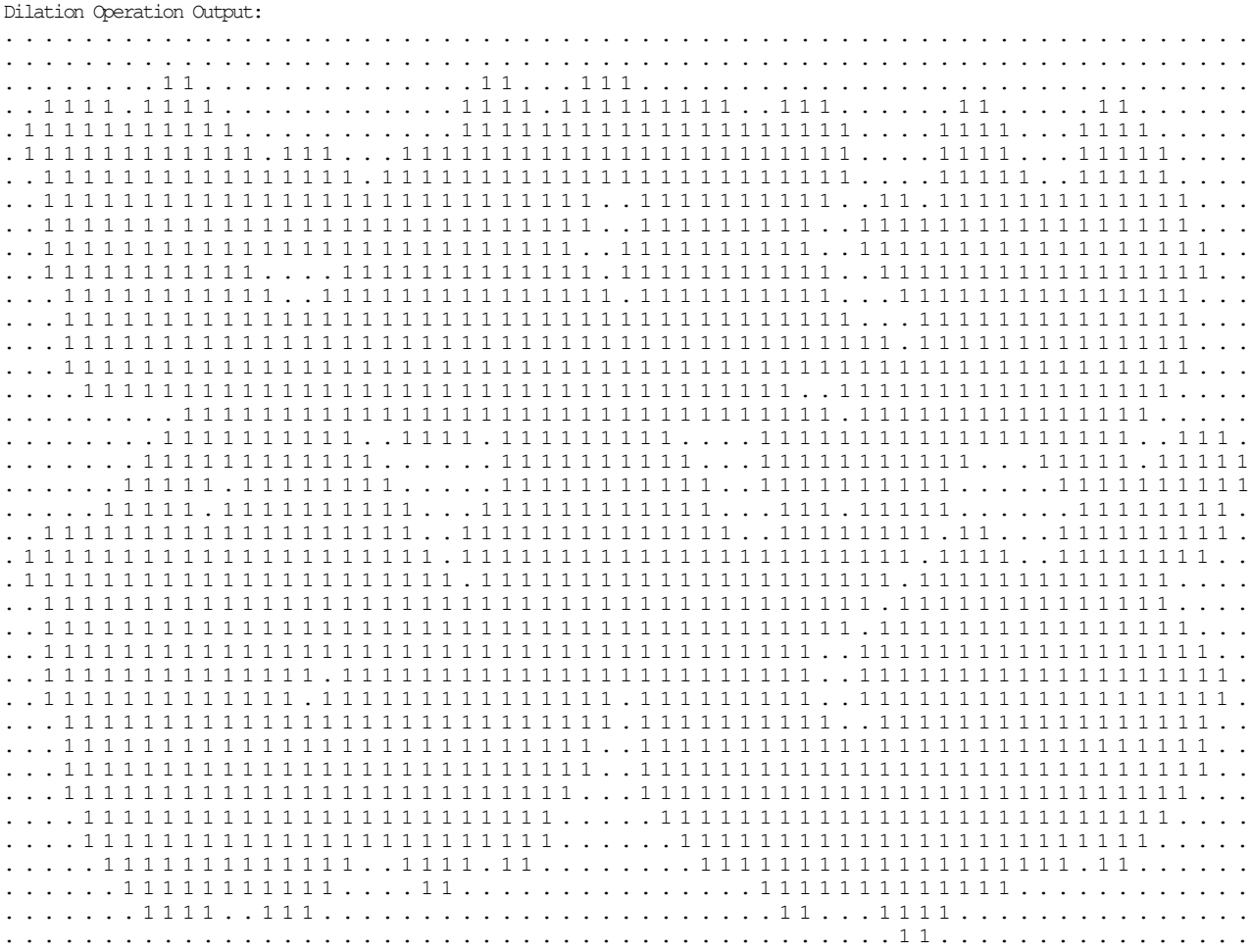
Image 2, Structured Element 3
****
Initial Zero Framed 2D Array:



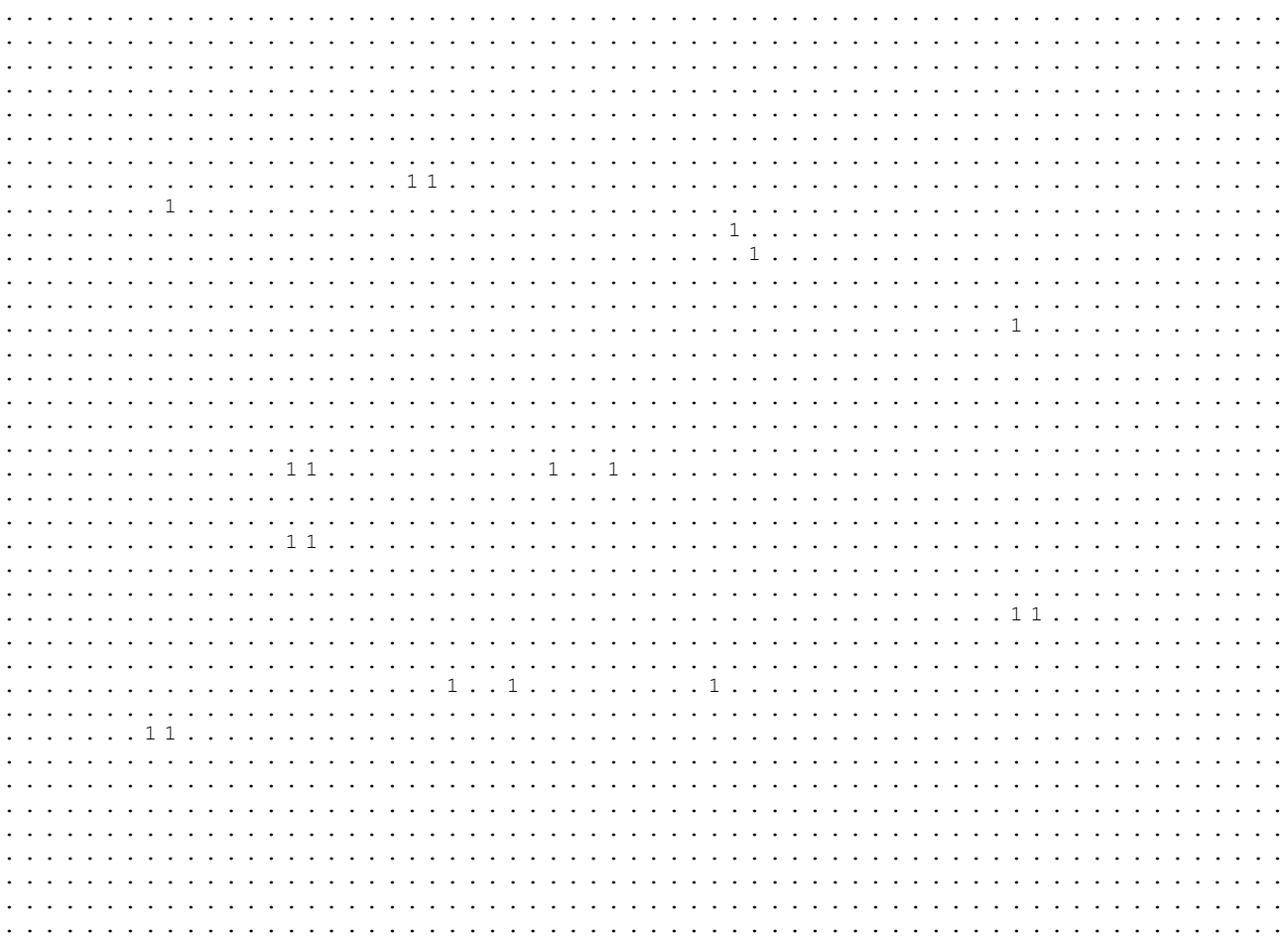
Initial Structured Element 2D Array:

```
. 1 1 .
1 1 1 1
1 1 1 1
. 1 1 .
```

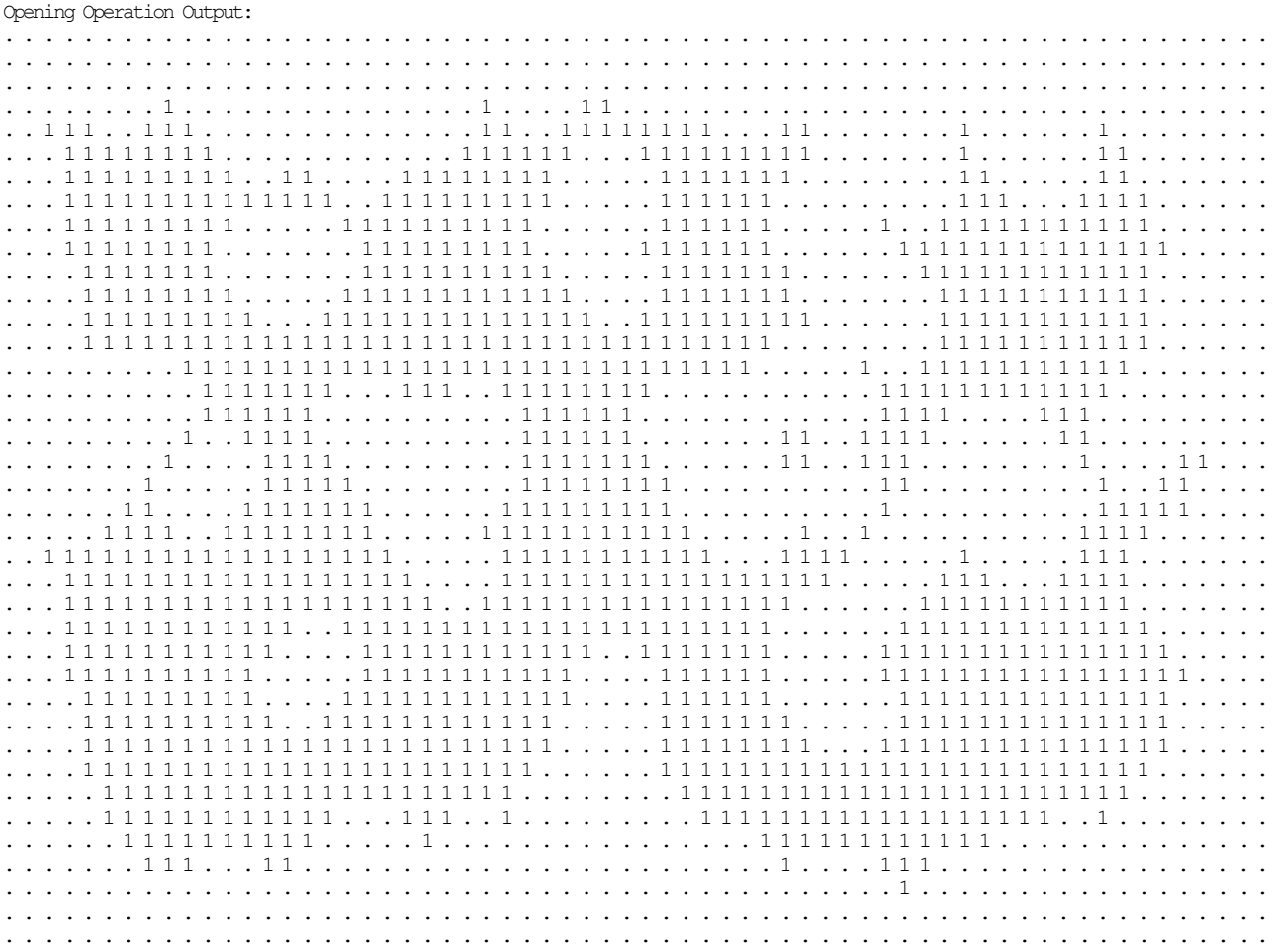
Dilation Operation Output:



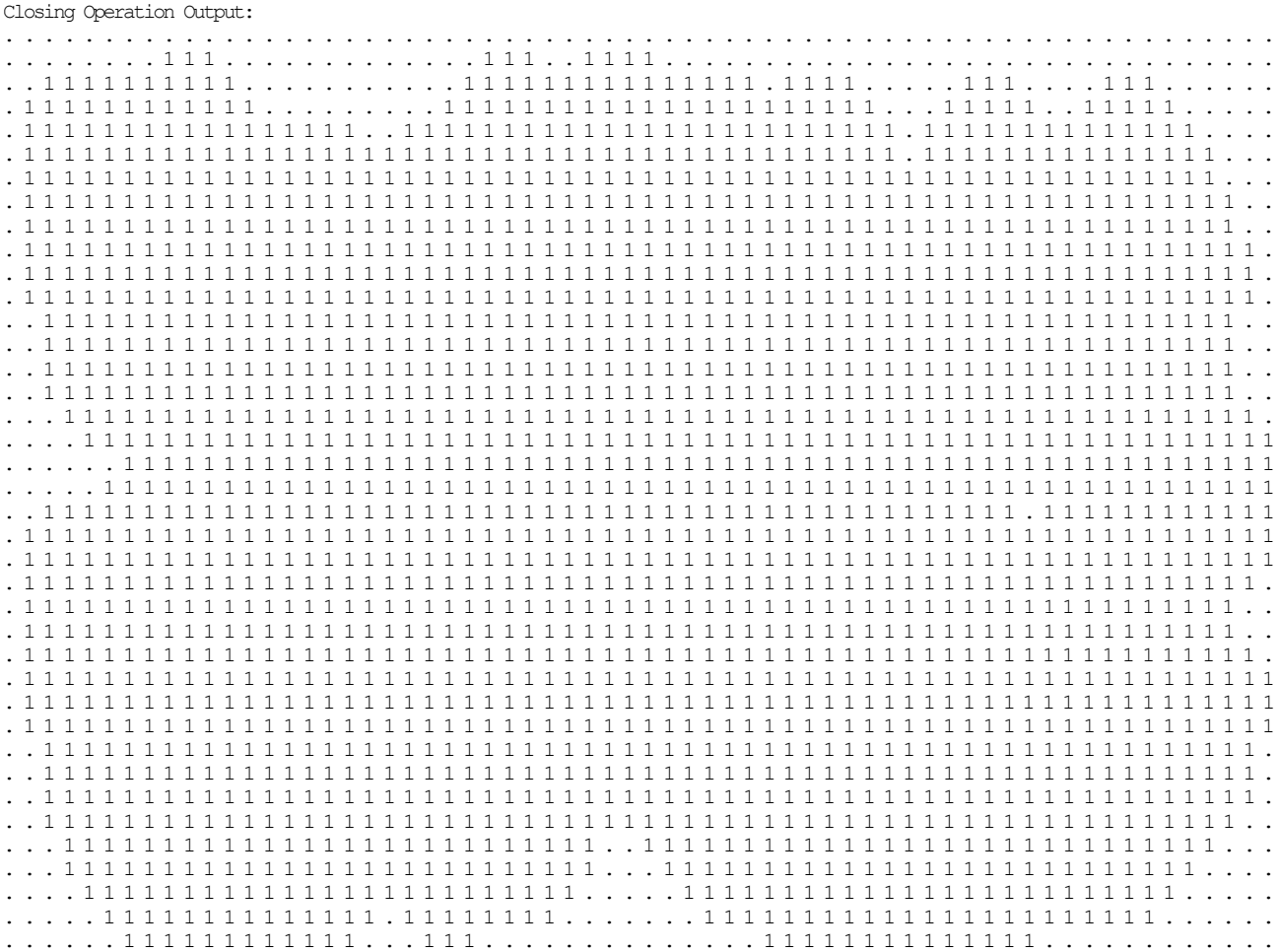
Erosion Operation Output:



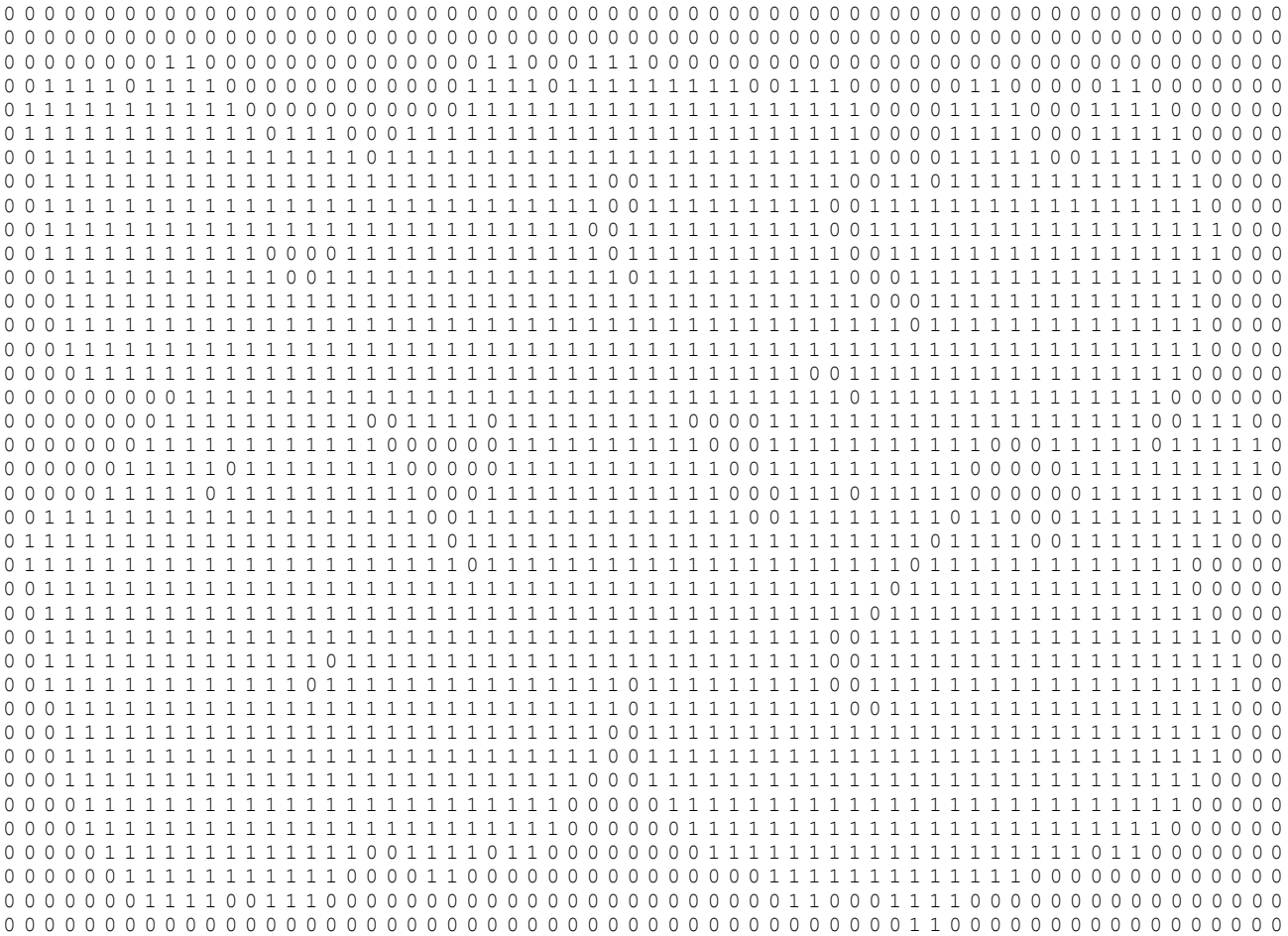
Opening Operation Output:



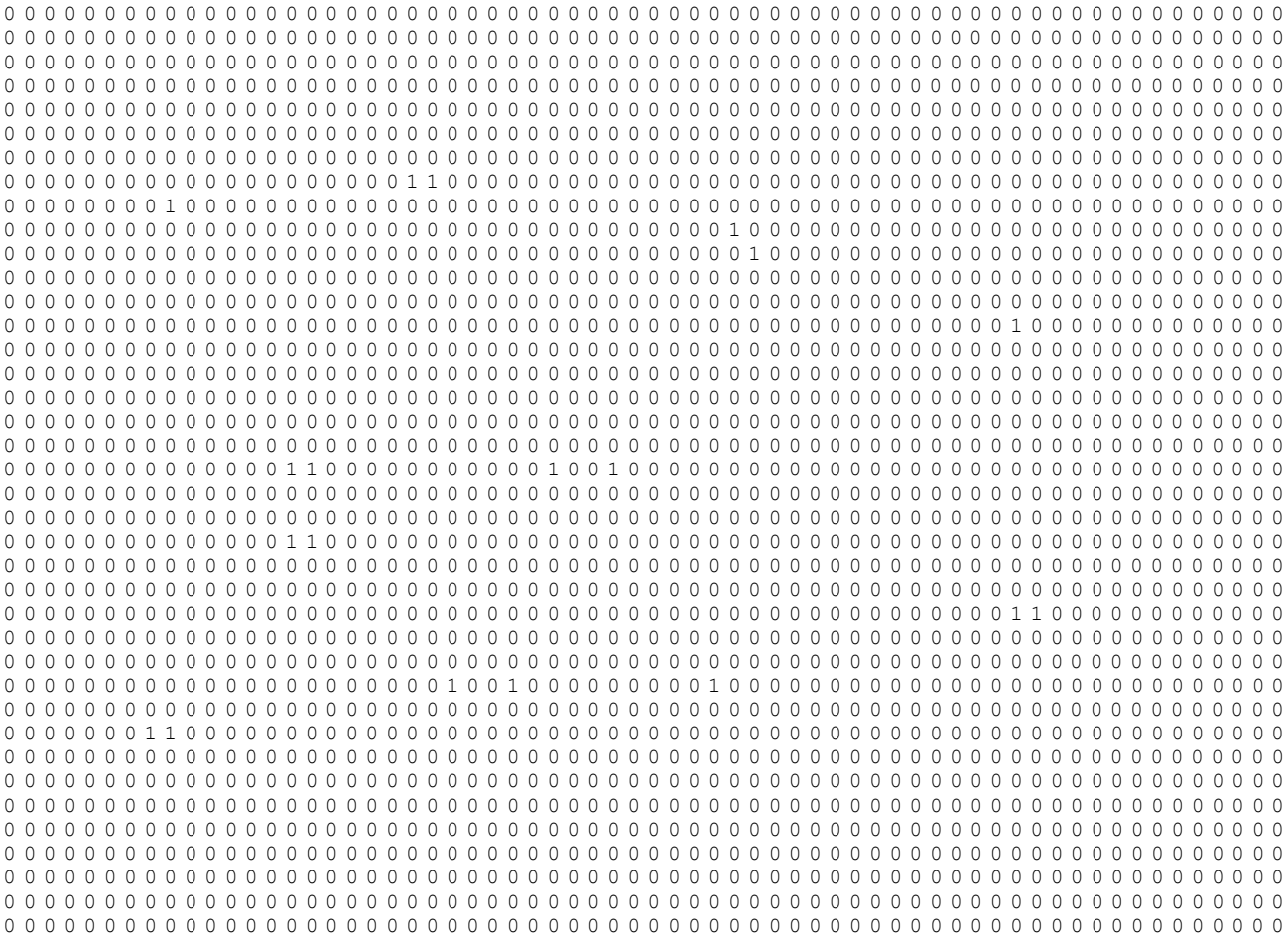
Closing Operation Output:



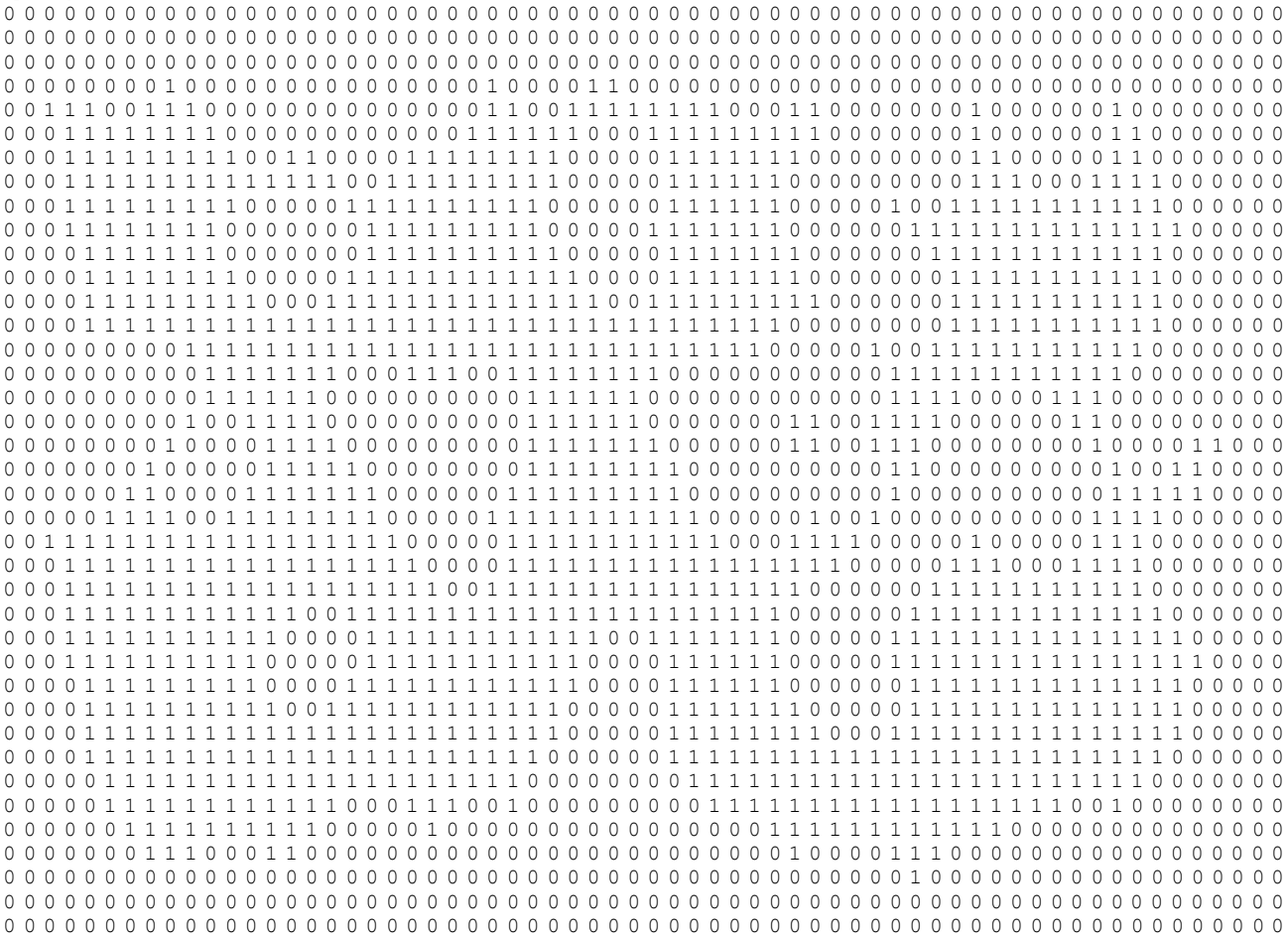
Dilation:



Erosion:



Opening:



Closing: