Name: Alif Rahi
Section: CSCI 381 - Computer Vision / Tues-Thurs 1:40-2:55pm
Project: 3
Due: Feb 27, 2023

Main algorithm steps:

step 0: imgFile, elmFile, outFile1, outFile2 ← open
step 1: numImgRows, numImgCols, imgMin, imgMax ← read from imgFile
numStructRows, numStructCols, structMin, structMax ← read from elmFile
rowOrigin, colOrigin ← read from elmFile

step 2: zeroFramedAry, structAry, morphAry, tempAry ← dynamically allocate. // see description in the above
step 3: zero2DAry(zeroFramedAry, rowSize, colSize)
step 4: loadImg (imgFile, zeroFramedAry)
step 5: imgReformat (zeroFramedAry, outFile1) // with caption.
prettyPrint (zeroFramedAry, outFile1) // with caption.

step 6: zero2DAry(structAry, numStructRows, numStructCols)
loadstruct (structFile, structAry)
prettyPrint (structAry, outFile1) // with caption.

step 7: basicOperations (zeroFramedAry, morphAry, structAry, tempAry, outFile1)
step 8: complexOperations (zeroFramedAry, morphAry, structAry, tempAry, outFile2)
step 9: close all files.

Source code

```
package RahiA_Project3;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;



public class RahiA_Project3_Main {
	static Scanner imgScan;
	static Scanner elmScan;
	static BufferedWriter out1;
	static BufferedWriter out2;


	static class Morphology{

		int numImgRows, numImgCols,imgMin,imgMax,numStructRows,numStructCols;
		int structMin, structMax, rowOrigin, colOrigin, rowFrameSize, colFrameSize;
```

```
	int extraRows, extraCols, rowSize, colSize;
	int[][] zeroFramedAry, morphAry, tempAry, structAry;

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


	}

	public void zero2DAry(int[][] arr, int rSize, int cSize) {
		for(int i=0; i<rSize; i++) {
			for(int j=0; j<cSize; j++) {
				Arrays.fill(arr[i], 0);
			}
		}
	}

	public void loadImg() {
		for(int i=rowOrigin; i<=numImgRows; i++) {
			for(int j=colOrigin; j<=numImgCols; j++) {
				zeroFramedAry[i][j] = imgScan.nextInt();
			}
		}


	}

	public void loadStruct() {
		for(int i=0; i<numStructRows; i++) {
			for(int j=0; j<numStructCols; j++) {
				structAry[i][j] = elmScan.nextInt();
```

```
                    }
                }


        }

public void imgReformat(int arr[][])
{
   String str = this.imgMax+"";
   int width = str.length();
   for (int r = 0; r < this.numImgRows+2; r++)
   {
      for (int c = 0; c < this.numImgCols+2; c++)
      {
        System.out.print(arr[r][c]+" ");
         str = arr[r][c]+"";
         int WW = str.length();
         while (WW < width)
         {
                  System.out.print(" ");
            WW++;
         }
      }
      System.out.println();
   }
   System.out.println();
}

        public void prettyPrint(int[][] arr, int r, int c, BufferedWriter out) {

             for (int i = 0; i < r; i++)
             {
                for (int j = 0; j < c; j++)
                {
                   if (arr[i][j] == 0)
                   {
                            try {
                                                        out.write(". ");
                                                    } catch (IOException e) {
                                                        e.printStackTrace();
                                                    }


                   }
                   else
                   {
                            try {
                                                        out.write("1 ");
                                                    } catch (IOException e) {
                                                        e.printStackTrace();
                                                    }

                   }
                }
                  try {
                                            out.write("\n");
```

```
                        } catch (IOException e) {
                                e.printStackTrace();
                        }

        }


}

public void basicOperations() {
        try {
                out1.write("Entering basicOperations method \n");
        } catch (IOException e2) {
                e2.printStackTrace();
        }

        zero2DAry(morphAry, rowSize, colSize);
        ComputeDilation(zeroFramedAry, morphAry, structAry);
        try {
                out1.write("Printing result of ComputeDilation \n");
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        prettyPrint(morphAry, rowSize, colSize, out1);


        zero2DAry(morphAry, rowSize, colSize);
        ComputeErosion(zeroFramedAry, morphAry, structAry);
        try {
                out1.write("Printing result of ComputeErosion \n");
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        prettyPrint(morphAry, rowSize, colSize, out1);




        zero2DAry(morphAry, rowSize, colSize);
        ComputeOpening(zeroFramedAry, morphAry, tempAry, structAry);
        try {
                out1.write("Printing result of ComputeOpening \n");
        } catch (IOException e1) {
                e1.printStackTrace();
        }
        prettyPrint(morphAry, rowSize, colSize, out1);


        zero2DAry(morphAry, rowSize, colSize);
        ComputeClosing(zeroFramedAry, morphAry, tempAry, structAry);
        try {
                out1.write("Printing result of ComputeClosing \n");
```

```
		} catch (IOException e1) {
			e1.printStackTrace();
		}
		prettyPrint(morphAry, rowSize, colSize, out1);

		try {
			out1.write("Exit basicOperations method \n");
		} catch (IOException e1) {
			e1.printStackTrace();
		}

	}

	public void complexOperations() {
		try {
			out2.write("Entering complexOperations method \n");
		} catch (IOException e2) {
			e2.printStackTrace();
		}

		zero2DAry(morphAry, rowSize, colSize);
		ComputeOpening(zeroFramedAry, morphAry,tempAry, structAry);
		try {
			out2.write("Pretty print the result of Opening \n");
		} catch (IOException e1) {
			e1.printStackTrace();
		}
		prettyPrint(morphAry, rowSize, colSize, out2);
		copyArys(morphAry, zeroFramedAry);



		zero2DAry(morphAry, rowSize, colSize);
		ComputeClosing(zeroFramedAry, morphAry,tempAry, structAry);
		try {
			out2.write("Pretty print the result of Opening follow by Closing. \n");
		} catch (IOException e1) {
			e1.printStackTrace();
		}
		prettyPrint(morphAry, rowSize, colSize, out2);
		copyArys(morphAry, zeroFramedAry);


		zero2DAry(morphAry, rowSize, colSize);
		ComputeClosing(zeroFramedAry, morphAry, tempAry,structAry);
		try {
			out2.write("Pretty print the result of Closing. \n");
		} catch (IOException e1) {
			e1.printStackTrace();
		}
		prettyPrint(morphAry, rowSize, colSize, out2);
		copyArys(morphAry, zeroFramedAry);
```

```
                zero2DAry(morphAry, rowSize, colSize);
                ComputeOpening(zeroFramedAry, morphAry, tempAry, structAry);
                try {
                        out2.write("Pretty print the result of Closing follow by Opening. \n");
                } catch (IOException e1) {
                        e1.printStackTrace();
                }
                prettyPrint(morphAry, rowSize, colSize, out2);
                copyArys(morphAry, zeroFramedAry);

                try {
                        out2.write("Exit complexOperations method \n");
                } catch (IOException e1) {
                        e1.printStackTrace();
                }

        }

        private void copyArys(int[][] morphAry, int[][] zeroFramedAry) {
                for(int i=0; i<morphAry.length; i++) {
                        for(int j=0; j<morphAry[i].length; j++) {
                                zeroFramedAry[i][j] = morphAry[i][j];
                        }
                }

        }

        private void ComputeDilation(int[][] inAry, int[][] outAry, int[][] structAry) {
                for(int i=rowFrameSize; i<rowSize; i++) {
                        for(int j=colFrameSize; j<colSize; j++) {
                                if(inAry[i][j] >0) {
                                        onePixelDilation(i,j,inAry, outAry, structAry);
                                }
                        }
                }

        }

        private void ComputeErosion(int[][] inAry, int[][] outAry, int[][] structAry) {
                for(int i=rowFrameSize; i<rowSize; i++) {
                        for(int j=colFrameSize; j<colSize; j++) {
                                if(inAry[i][j] >0) {
                                        onePixelErosion(i,j,inAry, outAry, structAry);
                                }
                        }
                }

        }

        private void ComputeOpening (int[][] zeroFramedAry, int[][] morphAry, int [][] tempAry,
int[][] structAry) {
                ComputeErosion(zeroFramedAry, tempAry, structAry);
                ComputeDilation(tempAry, morphAry, structAry);
        }
```

```
		private void ComputeClosing(int[][] zeroFramedAry, int[][] morphAry, int [][] tempAry, int[][]
structAry) {
			ComputeDilation(zeroFramedAry, tempAry, structAry);
			ComputeErosion(tempAry, morphAry, structAry);
		}

		private void onePixelErosion(int i, int j, int[][] inAry, int[][] outAry, int[][] structAry) {
			int iOffset = i - rowOrigin;
			int jOffset = j - colOrigin;
			boolean matchFlag = true;
			for(int ridx = 0; matchFlag && ridx < numStructRows; ridx++) {
				for(int cidx =0; matchFlag && cidx < numStructCols; cidx++) {
					if((iOffset + ridx) < outAry.length && (jOffset + cidx) <
outAry[i].length){
						if(structAry[ridx][cidx] > 0 && inAry[iOffset + ridx][jOffset +
cidx] <=0) {
							matchFlag = false;
						}
					}
				}
			}
			if(matchFlag == true) {
				outAry[i][j] = 1;
			}
			else {
				outAry[i][j] = 0;
			}

		}

		private void onePixelDilation(int i, int j, int[][] inAry, int[][] outAry, int[][] structAry) {
			int iOffset = i - rowOrigin;
			int jOffset = j - colOrigin;
			for(int ridx = 0; ridx < numStructRows; ridx++) {
				for(int cidx =0; cidx < numStructCols; cidx++) {
					if((iOffset + ridx) < outAry.length && (jOffset + cidx) <
outAry[i].length  && structAry[ridx][cidx] > 0) {
						outAry[iOffset + ridx][jOffset + cidx] = 1;
					}
				}
			}

		}

	}



	public static void main(String[] args) {
		File imgFile = new File(args[0]);
		File elmFile = new File(args[1]);
		File outFile1 = new File(args[2]);
		File outFile2 = new File(args[3]);
```

```
		try {
			out1 = new BufferedWriter(new FileWriter(outFile1));
			out2 = new BufferedWriter(new FileWriter(outFile2));
		} catch (IOException e1) {
			e1.printStackTrace();
		}
		Morphology morphology = null;
		int numImgRows = 0, numImgCols = 0,imgMin = 0,imgMax =0,numStructRows =
0,numStructCols = 0;
		int structMin, structMax, rowOrigin, colOrigin, rowFrameSize, colFrameSize;
		int extraRows, extraCols, rowSize, colSize;

		try {
			imgScan = new Scanner(imgFile);
			numImgRows = imgScan.nextInt();
			numImgCols = imgScan.nextInt();
			imgMin = imgScan.nextInt();
			imgMax = imgScan.nextInt();


		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}

		try {
			elmScan = new Scanner(elmFile);

			numStructRows = elmScan.nextInt();
			numStructCols = elmScan.nextInt();
			structMin = elmScan.nextInt();
			structMax = elmScan.nextInt();
			rowOrigin = elmScan.nextInt();
			colOrigin = elmScan.nextInt();
			rowFrameSize = numStructRows /2;
			colFrameSize = numStructCols /2;
			extraRows = rowFrameSize * 2;
			extraCols = colFrameSize * 2;
			rowSize = numImgRows + extraRows;
			colSize = numImgCols + extraCols;

			morphology = new Morphology(numImgRows, numImgCols, imgMin,imgMax,
numStructRows, numStructCols, structMin, structMax, rowOrigin, colOrigin, rowFrameSize, colFrameSize,
			 extraRows, extraCols, rowSize, colSize);

		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}

		morphology.zero2DAry(morphology.zeroFramedAry,  numImgRows, numImgCols);
		morphology.loadImg();
		morphology.imgReformat(morphology.zeroFramedAry);
		try {
			out1.write("Printing zero framed array \n");
		} catch (IOException e1) {
```

```
                    e1.printStackTrace();
                }
                morphology.prettyPrint(morphology.zeroFramedAry, numImgRows +2, numImgCols+2,
out1);



                morphology.zero2DAry(morphology.structAry, numStructRows, numStructCols);
                morphology.loadStruct();
                try {
                    out1.write("Printing struct array \n");
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
                morphology.prettyPrint(morphology.structAry, numStructRows, numStructCols, out1);

                morphology.basicOperations();
                morphology.complexOperations();
                try {
                    out2.close();
                    out1.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }


        }


}
```

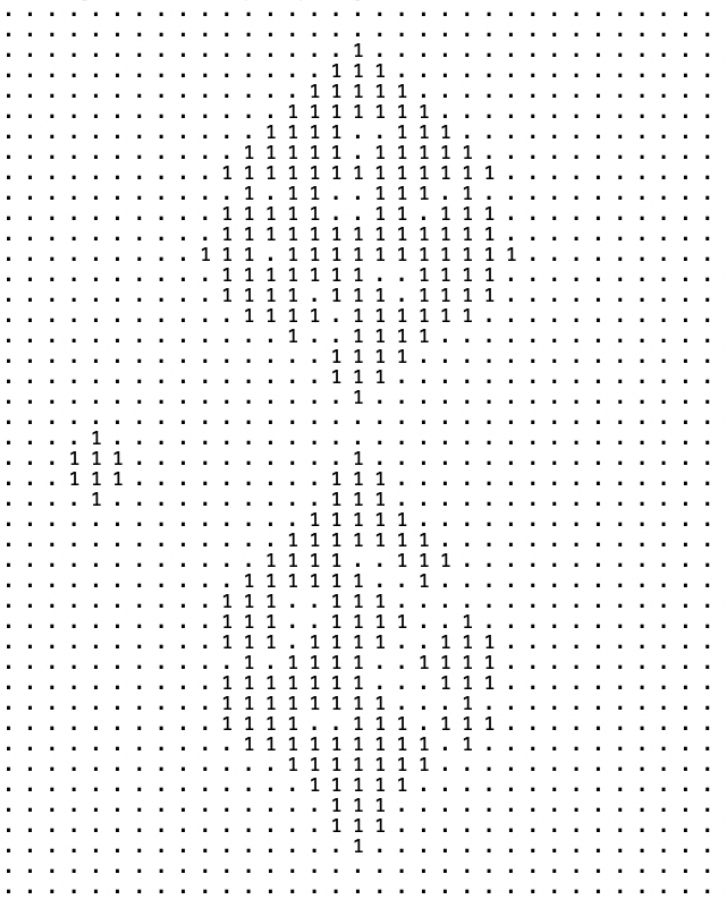
outFile1 and outFile2 from **Run1:**

outFile1:

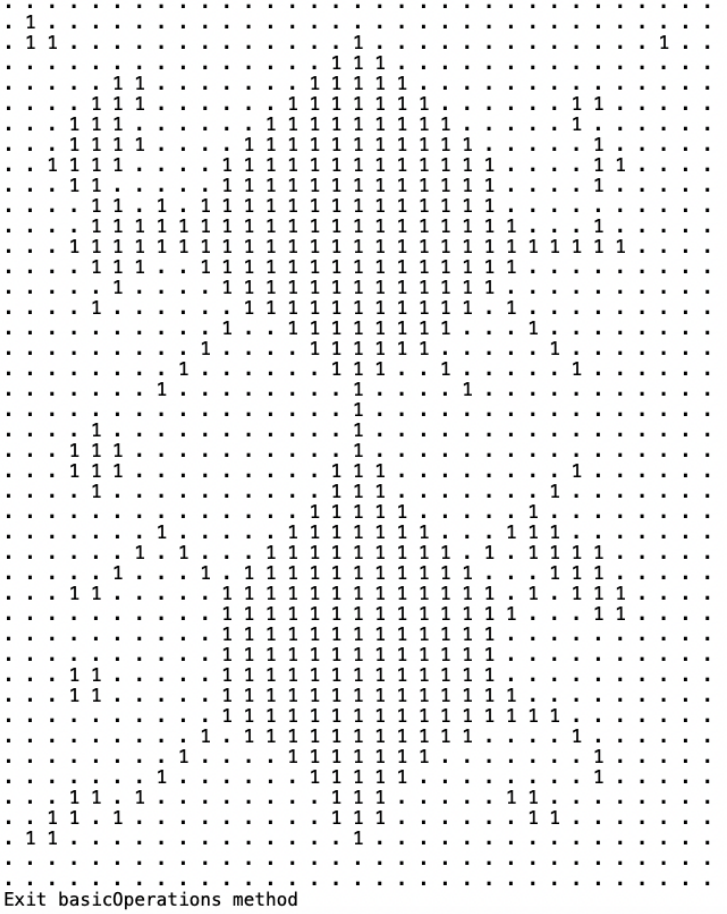
```
Printing zero framed array
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. 1 1 . . . . . . . . . . . . . 1 . . . . . . . . . . . . . 1 . .
. . . . . . . . . . . . . . . 1 1 1 . . . . . . . . . . . . . . .
. . . . . 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 . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Printing struct array
. 1 .
1 1 1
. 1 .
```

```
Entering basicOperations method
Printing result of ComputeDilation
. 1 . . . . . .
1 1 1 . . . . . 1 . . . . . 1 .
1 1 1 1 . . . . 1 1 1 . . . . 1 1 1 .
. 1 1 . . 1 1 . . . . . 1 1 1 1 1 . . . . . 1 . .
. . . . 1 1 1 1 1 . . . . 1 1 1 1 1 1 1 . . . . . 1 1 . . . . .
. . . 1 1 1 1 1 1 . . . . 1 1 1 1 1 1 1 1 1 . . . . 1 1 1 1 . . . .
. . 1 1 1 1 1 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 . . . . .
. . 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 . . . . .
. 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 . . .
. . 1 1 1 1 . 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 . . . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . . . 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . 1 1 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 . . . . . .
. . . . . 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 . . . . .
. . . . . 1 1 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . .
. . . . 1 . . 1 . . . . . 1 1 1 . . . 1 . . . . . .
. . . 1 1 1 . . . . . 1 1 1 . . . . . .
. . 1 1 1 1 1 . . . . . 1 1 1 . . . . . 1 . . . . . .
. . . 1 1 1 . . . . . 1 1 1 1 1 . . . . . 1 1 1 . . . . . .
. . . . 1 . . 1 . . . . . 1 1 1 1 1 1 1 . . . 1 1 1 . . . . . .
. . . . . 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 . . . . .
. . . . . 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . . 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . 1 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . 1 1 . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . .
. . . . . 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 . . . . 1 1 1 . . . .
. . . 1 1 . 1 1 1 . . . . 1 1 1 1 1 1 1 1 . . . 1 1 . 1 1 1 . . . .
. . 1 1 1 1 1 1 . . . . . 1 1 1 1 1 . . . 1 1 1 1 . 1 . . . . .
. 1 1 1 1 1 1 . . . . . 1 1 1 1 1 . . . . 1 1 1 1 . . . . . .
1 1 1 1 . 1 . . . . . 1 1 1 . . . . . 1 1 . . . . . .
. 1 1 . . . . . 1 . . . . . .
. . . . . .
Printing result of ComputeErosion
. . . . . .
. . . . . .
. . . . . .
. . . . . 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 . . 1 . . . . . .
. . . . . 1 1 . . . . 1 . . . . . .
. . . . . 1 1 1 . . . 1 1 1 . . . . . .
. . . . . 1 . 1 1 . . 1 1 1 . 1 . . . . . .
. . . . . 1 . . . . . 1 . . . . . .
. . . . . 1 . 1 . . . . . 1 . . . . . .
. . . . . 1 . 1 1 . . 1 1 . 1 1 . . . . . .
. . . . . 1 . . . 1 1 1 . . 1 1 1 1 . . . . . .
. . . . . 1 . 1 . 1 . . . . 1 1 . . . . . .
. . . . . 1 1 . . . 1 . . . 1 1 . . . . . .
. . . . . 1 . . . 1 . 1 . . . . . .
. . . . . 1 1 . . 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 1 . . 1 . . . . . .
. . . . . 1 1 . . . . 1 . . . . . .
. . . . . 1 . . 1 . . . . . .
. . . . . 1 . . . . 1 . . . . . .
. . . . . 1 . . . . 1 1 . . . . . .
. . . . . 1 . . . 1 1 . . . . 1 . . . . . .
. . . . . 1 1 . . . . 1 1 . . . . . .
. . . . . 1 . 1 1 1 . . . . . 1 . . . . . .
. . . . . 1 1 1 . . 1 . . . . . .
. . . . . 1 1 . . . . 1 . . . 1 . . . . . .
. . . . . 1 . . 1 1 1 . . . . . .
. . . . . 1 1 1 1 1 . . . . . .
. . . . . 1 1 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . 1 . . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
Printing result of ComputeOpening
```



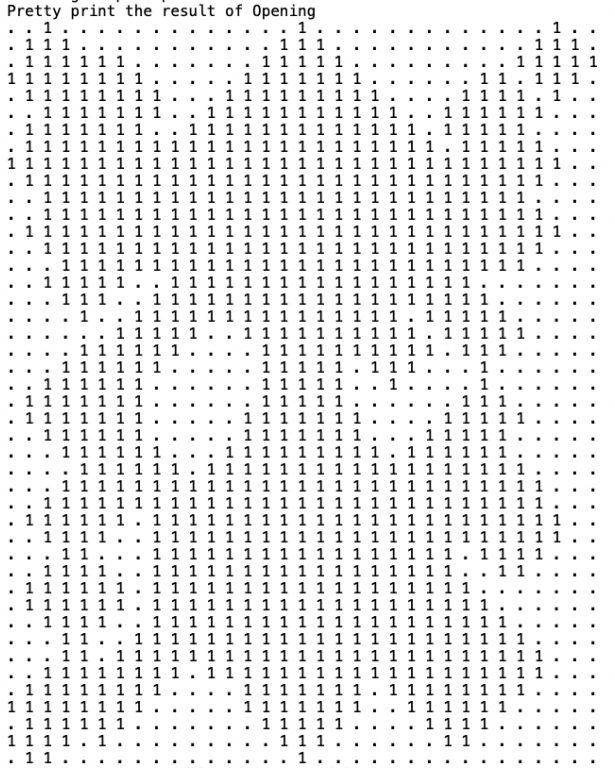
```
Printing result of ComputeClosing
```



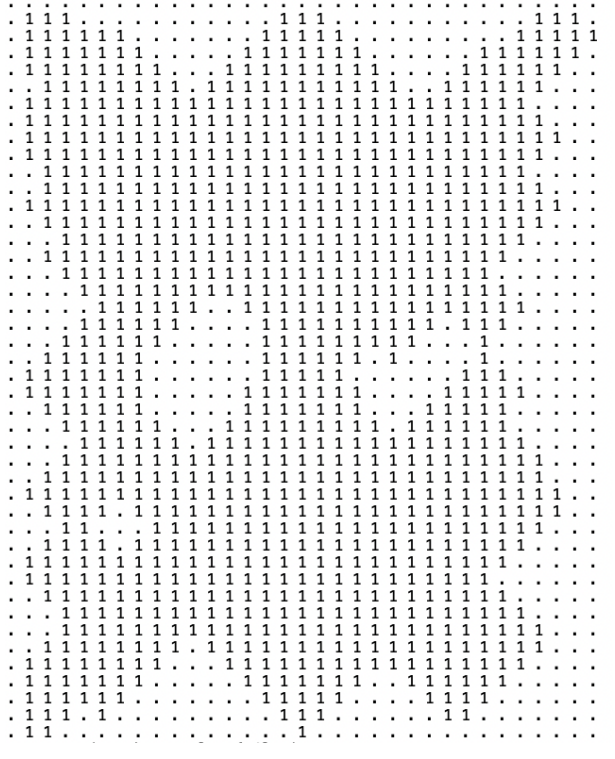
```
Exit basicOperations method
```

outFile2:

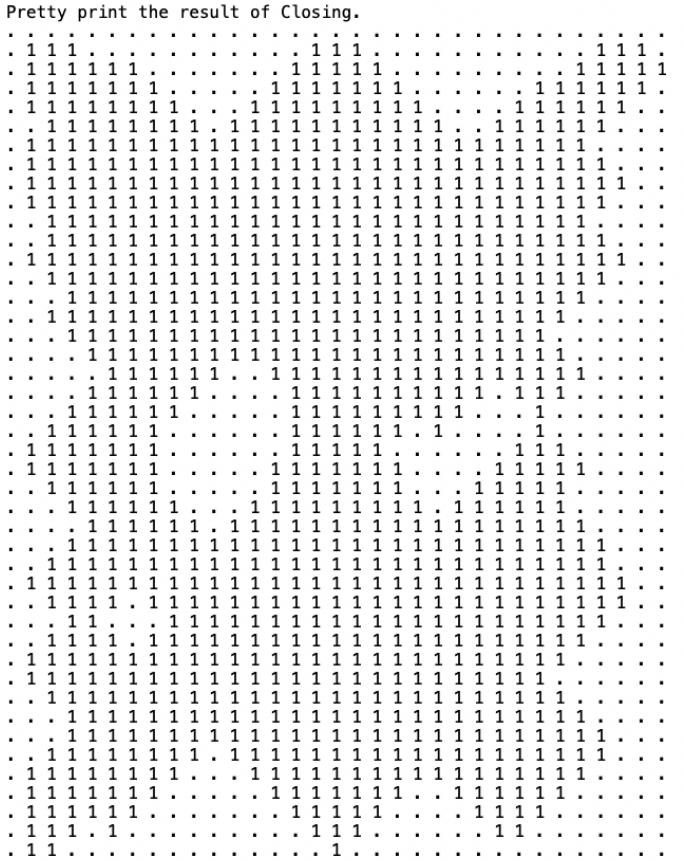
```
Entering complexOperations method
Pretty print the result of Opening
```



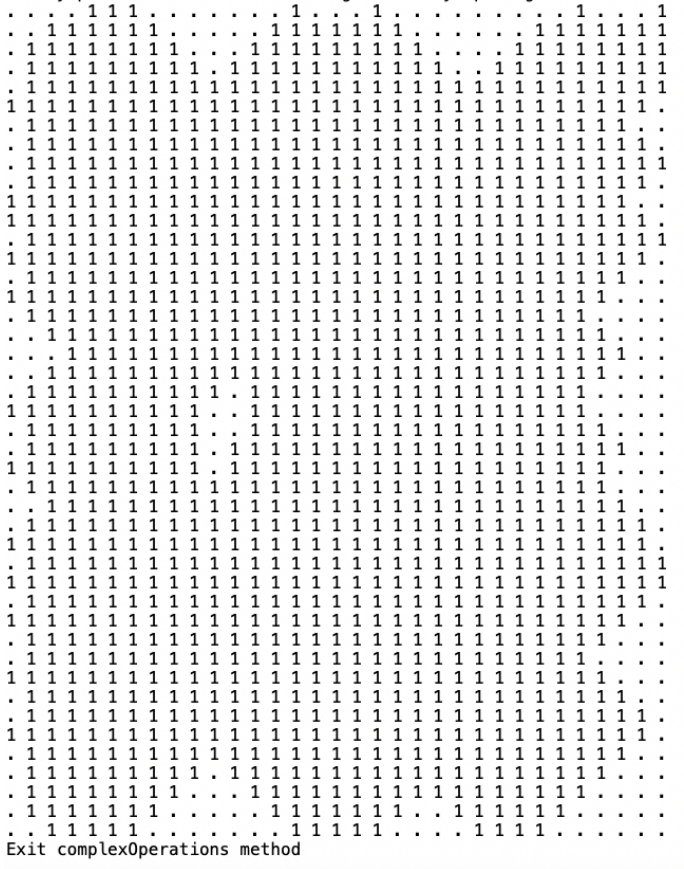
```
Pretty print the result of Opening follow by Closing.
```



Pretty print the result of Closing.



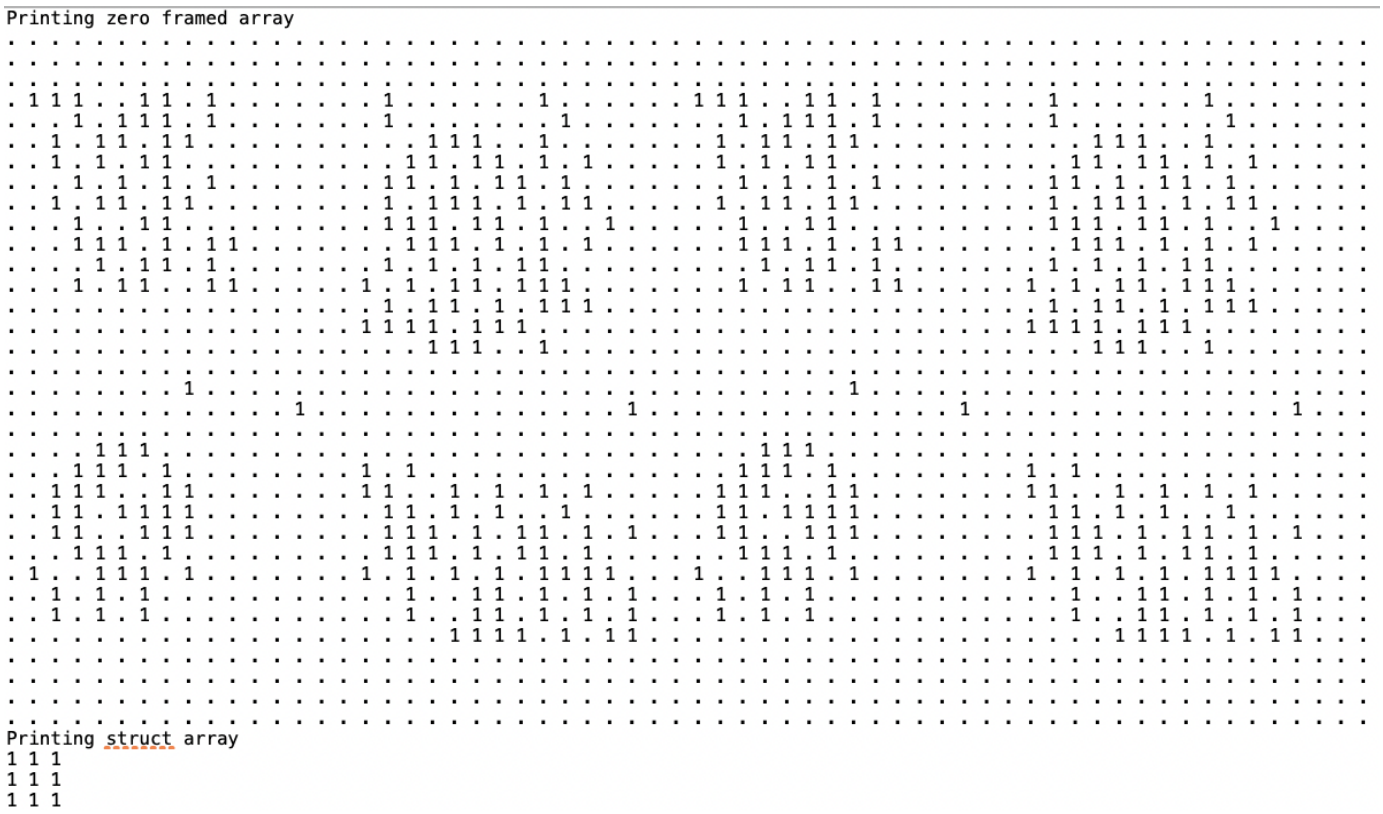
Pretty print the result of Closing follow by Opening.



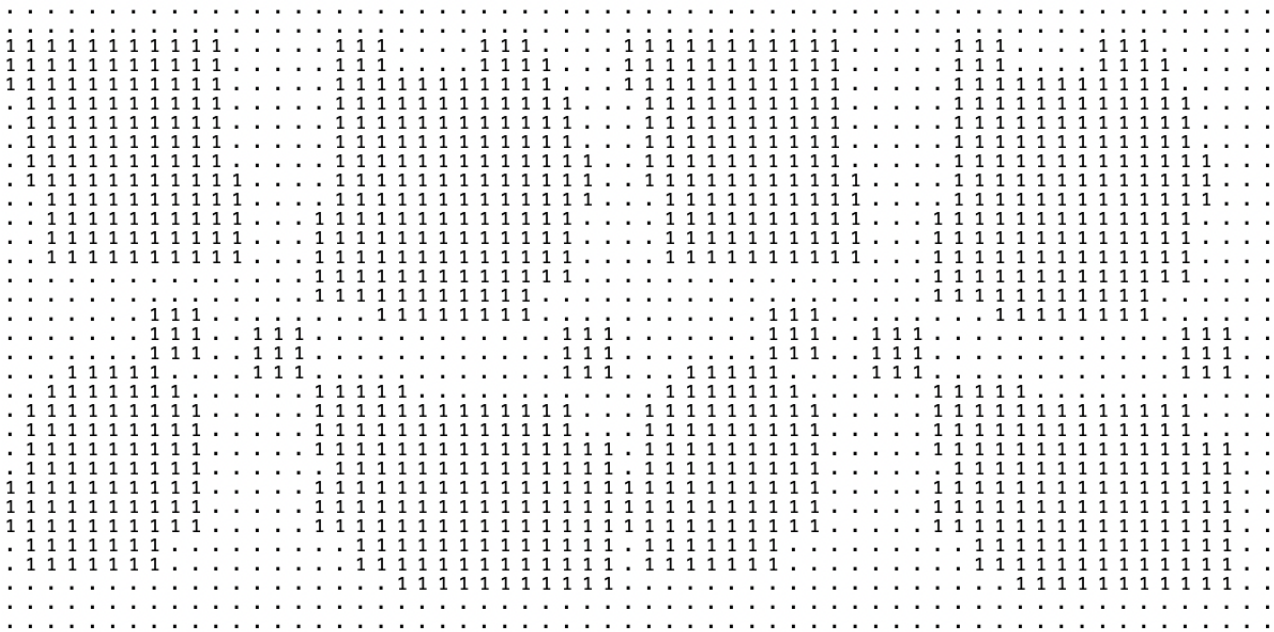
Exit complexOperations method

outFile1 and outFile2 from **Run2:**

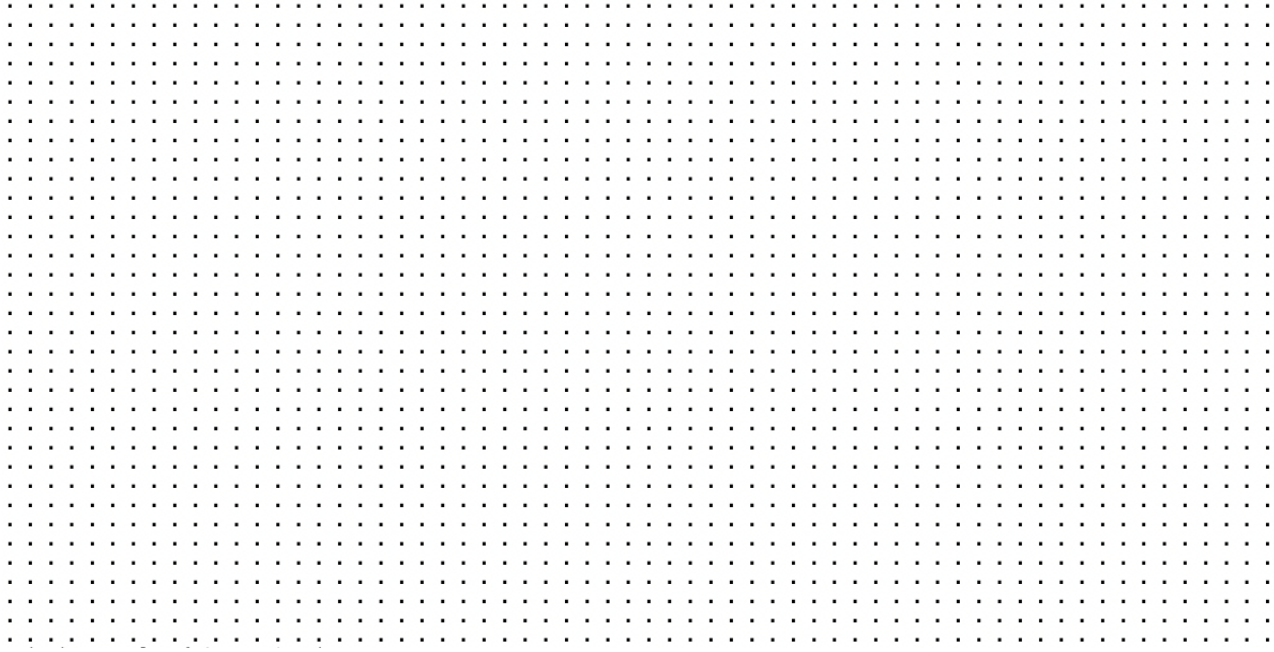
outFile1:



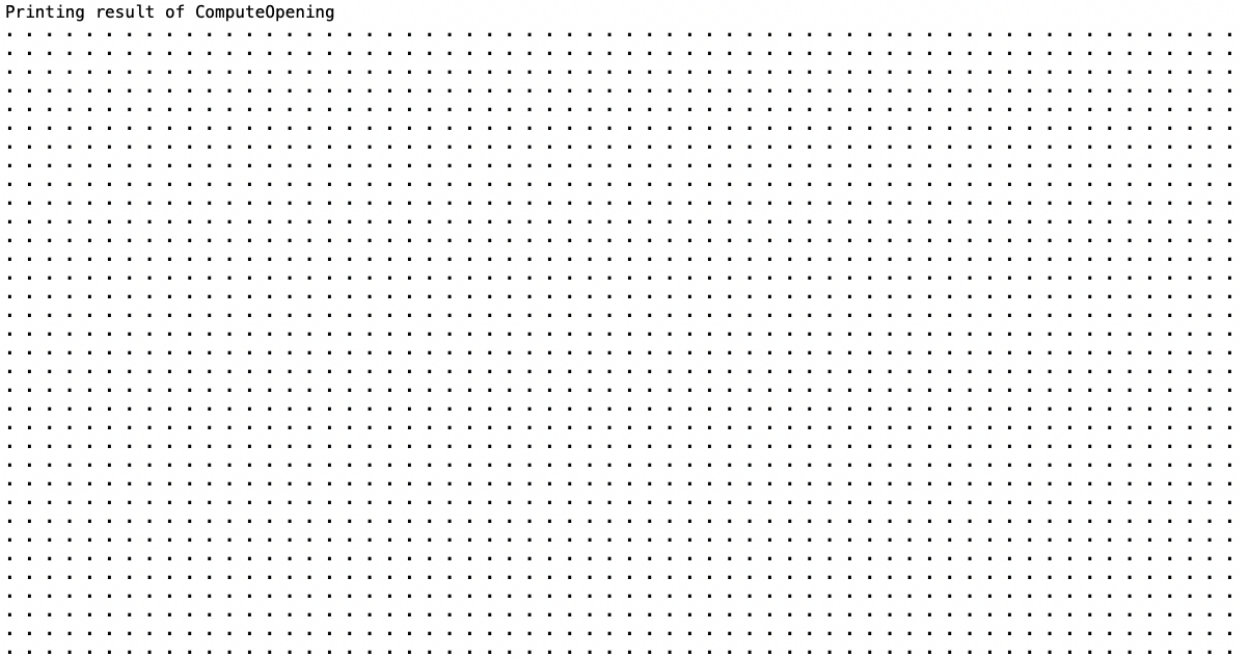
Entering basicOperations method
Printing result of ComputeDilation



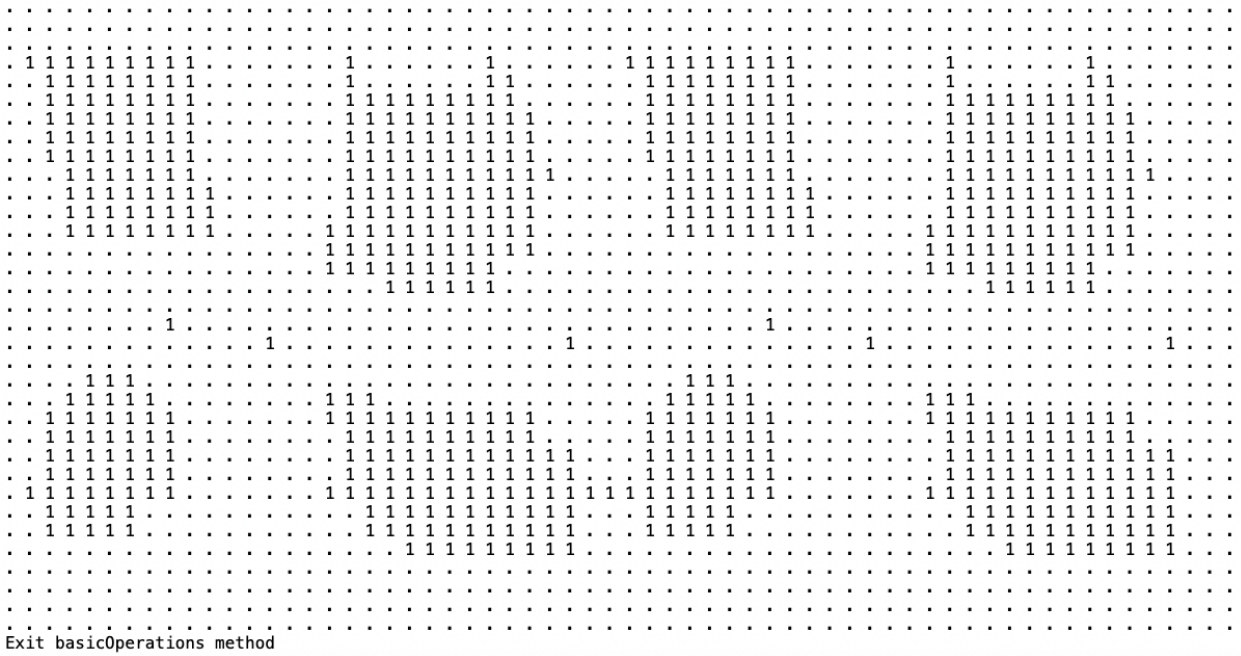
Printing result of ComputeErosion



Printing result of ComputeOpening



Printing result of ComputeClosing

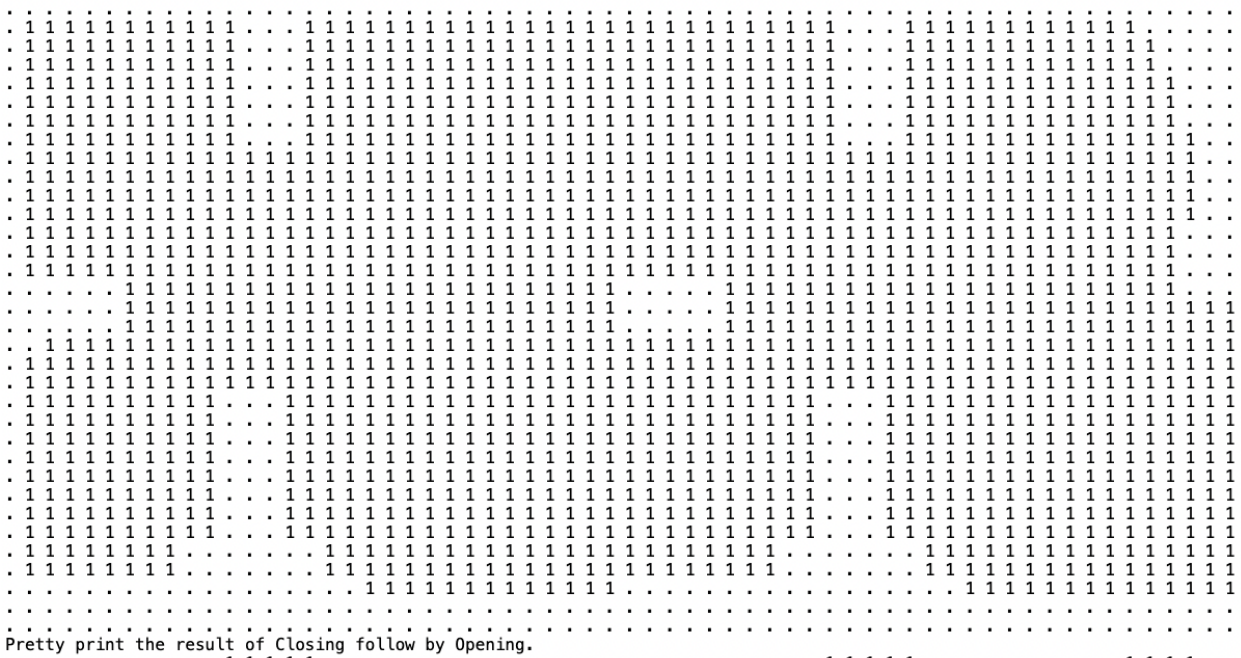


Exit basicOperations method

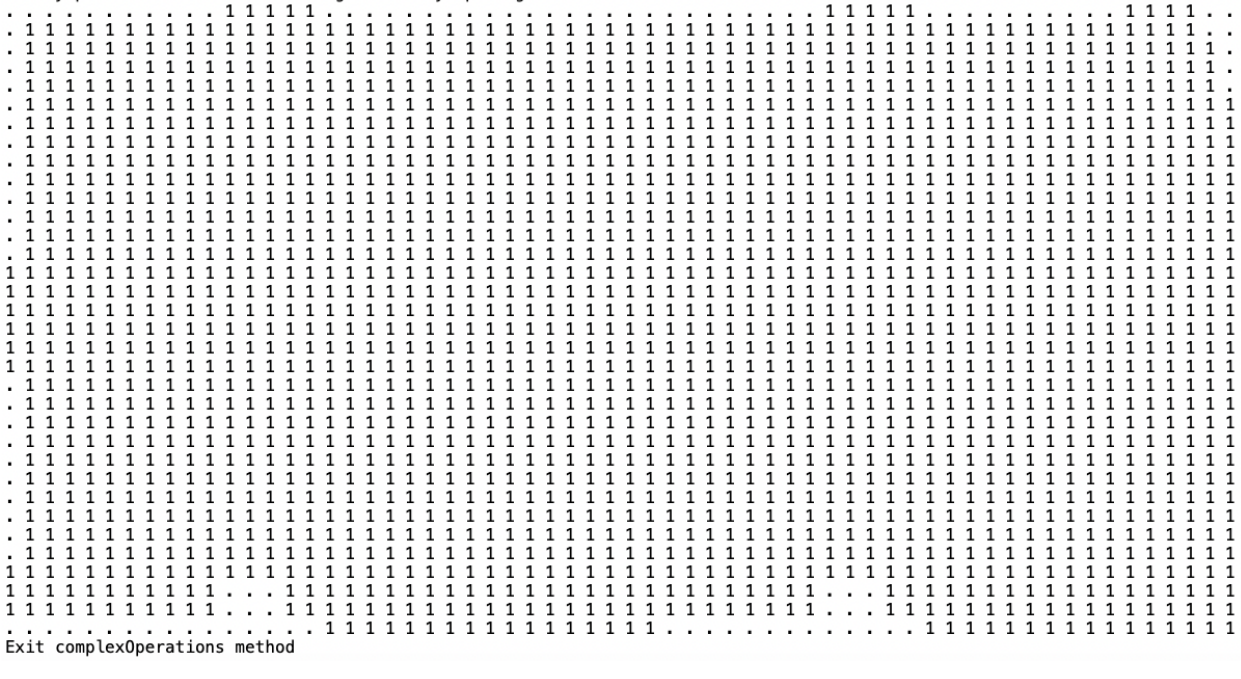
outFile2:

```
Entering complexOperations method
Pretty print the result of Opening
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 . . 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 . . 1 1 1 1 1 . . . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 . . 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 . . 1 1 1 1 1 1 . . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . . . . . 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . 1 1 1 1 1 .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Pretty print the result of Opening follow by Closing.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 . . . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . 1 1 1 1 1 1 1 1 1 1 1 1 1 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Pretty print the result of Closing.



Pretty print the result of Closing follow by Opening.



Exit complexOperations method