CV Project 8: ChainCode C++ Student: Zeehan Rahman

Project Due Date: 11/26/2021

Algorithmic Steps:

1. Open the input file, from argv[1]. Assign the correct values to the variables: numrows, numcols, minval, maxval
2. Open the 1st output file with the name as the name of input file + "_chaincode.txt". Similarly open the 2nd output file with the name as the name of the input file + "boundary.txt". Lastly, open the 3rd output file with the name as the name of the input file + "_decompressed.txt"
3. The constructor of the class ChainCode dynamically allocates the arrays: imgarray and reconstructarray with numrows + 2 and numcols + 2 as the dimensions. Call the method setZero once for each array that assigns the value 0 to all the elements of both the arrays.
4. Call the function loadimage that transfers all the contents of input file to the imagearray
5. Call the function reformatprettyprint that ouputs the contents of the imagerarray to the output file 1
6. Call the function getchaincode that generates the chain code for the given image and writes it to the chaincode file
7. Close and reopen the chaincode file
8. Call the function constructBoundary that constructs the boundary of the image by reading the chaincode file
9. Call the function reformartprettyprint that transfers all the contents of the reconstructarray to the input_boundary.txt output file
10. Call the function fillObject that fills the boundary of the object
11. Call the function reformatprettyprint that transfers the contents of the recontructarray to the input.txt_decompressed.txt
12. Finally, close all files

Source Code

//main

```
# include<fstream>
# include <iostream>
#include <string>
#include "ChainCode.cpp"

using namespace std;

int main(int argc, char** argv)
{
	if (argc != 2)
	{
		std::cout << "Wrong number of arguments\n";
		std::cout << "argc = " << argc << std::endl;
		exit(0);
	}
	int numRows = 0, numCols = 0, minval = 0, maxval = 0;
	ifstream inFile, compressedtxt;
```

```
	inFile.open(argv[1]);
	if (inFile.fail()) cout << "Error in opening " << argv[1] << "\n";
	inFile >> numRows; inFile >> numCols; inFile >> minval; inFile >> maxval;
	std::string header = to_string(numRows) + " " + to_string(numCols) + " " + to_string(minval) + " " +
to_string(maxval);

	std::string subname = "_chaincode.txt";
	std::string subname2 = "_boundary.txt";
	std::string subname3 = "_decompressed.txt";

	ofstream outFile(argv[1] + subname), outFile2(argv[1] + subname2), outFile3(argv[1] + subname3);

	ChainCode obj(numRows, numCols, minval, maxval, header);


	obj.loadImage(inFile);

	obj.GetChainCode(outFile); outFile.close(); compressedtxt.open(argv[1] + subname);

	obj.constBound(compressedtxt);

	obj.reformatPrettyPrint(outFile2, "Boundary:");

	obj.FillObject();

	//obj.showArray(obj.imgary, numRows, numCols);



	obj.reformatPrettyPrint(outFile3, "Decompressed File:");









	inFile.close();
	outFile.close();
	outFile2.close();


	return 0;
}
```

//ChainCode Class

```
#pragma once
#include <fstream>
# include <iostream>
#include <string>
#include <sstream>
#include "Point.cpp"
class ChainCode
{
```

```
public: int numRows, numCols, imgMin, imgMax, label, lastz = 4, chaindir =
0; public: Point coset[8], NeighCoord[8];
public: std::string header;
public: int Ztable[8] = { 6, 0, 0, 2, 2, 4, 4, 6 };
public: int** imgary;
public: int** reconstructary;

public: ChainCode(int nr, int nc, int min, int max, std::string head)
{
	header = head;
	numRows = nr, numCols = nc, imgMax = max, imgMin = min;

	imgary = new int* [numRows + 2];
	reconstructary = new int* [numRows + 2];

	Point obj(0, 1); coset[0] = obj; obj.setPoint(-1, 1); coset[1] = obj;
	obj.setPoint(-1, 0); coset[2] = obj; obj.setPoint(-1, -1); coset[3] = obj;
	obj.setPoint(0, -1); coset[4] = obj; obj.setPoint(1, -1); coset[5] = obj;
	obj.setPoint(1, 0); coset[6] = obj; obj.setPoint(1, 1); coset[7] = obj;



	for (int i = 0; i < numRows + 2; i++)
	{
		imgary[i] = new int[numCols + 2];
		reconstructary[i] = new int[numCols + 2];
	}

	setZero(reconstructary); setZero(imgary);
}

public:void setZero(int** ary)
{
	for (int i = 0; i < numRows + 2; i++)
	{
		for (int j = 0; j < numCols + 2; j++)
		{
			ary[i][j] = 0;
		}
	}
}

public: void loadImage(std::ifstream& inFile)
{

	for (int i = 1; i < numRows + 1; i++)
	{
		for (int j = 1; j < numCols + 1; j++)
		{
			std::string st;

			inFile >> st;

			std::stringstream str(st);
```

```
                str >> imgary[i][j];
            }
        }
}

public: void GetChainCode(std::ofstream& ccf)
{
        ccf << numRows << " " << numCols << " " << imgMin << " " << imgMax << "\n";
        Point currentP(0, 0);
        int startrow = 0, startcol = 0;
        bool found = false;

        for (int i = 1; (i < numRows + 1 && !found); i++)
        {

                for (int j = 1; (j < numCols + 1 && !found); j++)
                {
                        if (imgary[i][j] > 0)
                        {
                                found = true;
                                label = imgary[i][j];

                                startrow = i;
                                startcol = j;
                                ccf << imgary[i][j] << " " << i << " " << j << " ";

                                currentP.setPoint(i, j);

                                break;
                        }
                }

        }


        do
        {
                lastz = (lastz + 1) % 8;
                //ccf<<"lastz = " << lastz << " ";
                chaindir = findNextP(currentP, lastz);
                ccf << chaindir << " ";

                currentP.setPoint(NeighCoord[chaindir].row, NeighCoord[chaindir].col);

                if (chaindir > 0)
                {
                        lastz = Ztable[(chaindir - 1) % 8]; //ccf << "cdr = " << chaindir << " ";
                }
                else
                {
                        lastz = Ztable[7];
                }
```

```
	} while (currentP.row != startrow || currentP.col != startcol);

	ccf << -1;
	ccf << "\n";
}

public: int findNextP(Point P, int lz)
{
	LoadNeighbours(P);
	int loop = 0;
	int index = lz;

	while (loop<7)
	{
		int irow = NeighCoord[index].row;
		int icol = NeighCoord[index].col;

		if (imgary[irow][icol] == label)
			return index;
		index = (index + 1) % 8;
		loop++;
	}
	return -1;
}

public: void LoadNeighbours(Point P)
{
	int row = P.row + coset[0].row; int col = P.col +
	coset[0].col; Point np(row, col);
	NeighCoord[0] = np;

	row = P.row + coset[1].row; col = P.col + coset[1].col;
	np.setPoint(row, col);
	NeighCoord[1] = np;

	row = P.row + coset[2].row; col = P.col + coset[2].col;
	np.setPoint(row, col);
	NeighCoord[2] = np;

	row = P.row + coset[3].row; col = P.col + coset[3].col;
	np.setPoint(row, col);
	NeighCoord[3] = np;

	row = P.row + coset[4].row; col = P.col + coset[4].col;
	np.setPoint(row, col);
	NeighCoord[4] = np;

	row = P.row + coset[5].row; col = P.col + coset[5].col;
	np.setPoint(row, col);
	NeighCoord[5] = np;

	row = P.row + coset[6].row; col = P.col + coset[6].col;
	np.setPoint(row, col);
	NeighCoord[6] = np;
```

```
	row = P.row + coset[7].row; col = P.col + coset[7].col;
	np.setPoint(row, col);
	NeighCoord[7] = np;

}

public: void constBound(std::ifstream& cmtxt)
{
	int nr = 0, nc = 0, mv = 0, maxv = 0, label = 0, srow = 0, scol = 0, dir = 0;

	cmtxt >> nr; cmtxt >> nc; cmtxt >> mv; cmtxt >> maxv;
	cmtxt >> label; cmtxt >> srow; cmtxt >> scol;

	int bounlabel = label;

	reconstructary[srow][scol] = bounlabel;

	int currentrow = srow, currentcol = scol;


	int irow = 0, icol = 0;
	do
	{
		cmtxt >> dir;
		if (dir == -1)
		{

			break;
		}
		int irow = currentrow + coset[dir].row;
		int icol = currentcol + coset[dir].col;

		reconstructary[irow][icol] = bounlabel;

		currentrow = irow; currentcol = icol;

	} while (irow != srow || icol != scol);
}
public: void reformatPrettyPrint(std::ofstream& outfile, std::string caption)
{
	outfile << caption +"\n\n"+header+ "\n";

	for (int i = 1; i < numRows + 1; i++)
	{
		for (int j = 1; j < numCols + 1; j++)
		{
			if (reconstructary[i][j] > 0) { outfile << reconstructary[i][j] << " "; }
			else { outfile << "0 "; }
		}
		outfile << "\n";
	}

}

public: void FillObject()
{
```

```
for (int i = 1; i < numRows + 1; i++)
{
	for (int j = 1; j < numCols + 1; j++)
	{
		if (reconstructary[i][j] > 0)
		{


			if(reconstructary[i][j+1] == 0)
			{

				int k = 0;

				do
				{
					j++;
						bool norightpixel = false;

					if (reconstructary[i - 1][j] == 0) //top test fail
					{//change all the previous 1s (until the left bpixel) to 0

					 for (int k2 = 1; k2 < k + 1; k2++)
					{
						 reconstructary[i][j - k2] = 0;
					}
					//skip all the right pixels to 0 until a right bpixel

					 while (reconstructary[i][j+1] == 0)
					{
						j++;
						if (j == numCols)//no right bpixel found
						{
							norightpixel = true;
							break;
						}
					}
					if (!norightpixel)
					{
						continue;
					}
					else break;

					}
					else//top test pass
					{//bottom test

						bool bpixelfound2 = false;
					 for (int currow = i+1; currow < numRows + 1; currow++)
					{
						if (reconstructary[currow][j] > 0)
						{
							bpixelfound2 = true; break; //bottom test pass
						}
					}
					bool bpixelfound3 = false;
					if (bpixelfound2) //bottom test pass
					{
```

```
                                                for (int curcol = j+1; curcol < numCols + 1; curcol++)
                                                {
                                                       if (reconstructary[i][curcol] > 0)
                                                       {
                                                              bpixelfound3 = true; break; //right test pass
                                                       }
                                                }
                                                if (bpixelfound3)//right pass(final pass)
                                                {
                                                       reconstructary[i][j] = 1;

                                                       k++;
                                                }
                                         }



                                               }

                                               if (norightpixel) break;
                                        } while (reconstructary[i][j+1] == 0);


                                 }
                                 else
                                 {
                                        continue;
                                 }
                          }
                   }

      }
}



public:void showArray(int** ary, int nr, int nc)
{
              for (int i = 1; i < nr + 1; i++)
              {
                     for (int j = 1; j < nc + 1; j++)
                     {
                            std::cout << ary[i][j] << " ";
                     }
                     std::cout << std::endl;
              }
              std::cout << "\n\n\n";
}



};

//Point
#pragma once
```

```
class Point
{
public:int row, col;


public: Point(int r, int c)
{
	row = r;
	col = c;
}
public: Point()
{
}

public: void setPoint(int r, int c)
{
	row = r;
	col = c;
}


};
```

Output

//Image 1

```
20 40 0 5
0000000000000000000000000000000000000000000
0000000005000000055500000000000500000000000
0000000055500000055500000000005550000000000
0000000555550000055500000000055555000000000
0000005555550000055500000000555555000000000
0000005555550000055500000000555555000500000
0000005555550000055500000000555555000500000
0000005555550050055500000000555555000500000
0000005555550050055500005000555555000500000
0000005555550050055500005000555555000500000
0000005555555555555555555555555555000500000
0000005555555555555555555555555555000500000
0000005555550000055500000000555555000500000
0000005555550000055500000000555555000500000
0000005555550000055500000000555555000500000
0000005555550000055500000000555555000000000
0000000555000000055500000000055500000000000
```

ChainCode File:

```
20 40 0 5
5 2 1 0 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 1 1 2 2 2 1 0 0 0 0 7 6 6 6 6 6 0 0 2 2 2 2 2 1 0 0 0 0 0 0 0 7 6 6 6 6 7 7 1 1 2 2 2 2 2 1 0 0 7 6 6 6 6 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 5 4 4 3 2 2 2 2 2 2 2 3 3 5 5 6 6 6 6 6 6 5 4 3 3 2 6 5 5 4 3 3 2 2 2 2 2 2 2 2 4 4 6 6 6 6 6 6 6 5 4 3 2 2 6 6 5 4 3 2 2 2 2 2 2 3 3 -1
```

Boundary:

```
20 40 0 5
0000000000000000000000000000000000000000000
0000000005000000055500000000000500000000000
0000000005050000005050000000005050000000000
0000000050005000005050000000050005000000000
0000000500050000050500000000500050000000000
0000000500050000050500000000500050005000000
0000000500050000050500000000500050005000000
0000000500050050050500000000500050005000000
0000000500050050050500005000500050005000000
0000000500050050050500005000500050005000000
0000000500000550550005005050050005000500000
0000000500000000000005500005500000555500000
0000000500000000000000000000000000500050000
```

0 0 0 0 0 0 0 5 0 0 0 0 5 5 5 5 5 0 0 0 5 5 5 5 5 5 5 5 0 0 0 0 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0 5 0 0 0 5 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 0 5 0 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0 0 5 0 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0

Decompressed File:

20 40 0 5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 5 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 5 0 0 5 5 5 0 0 0 0 5 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 5 0 0 5 5 5 0 0 0 0 5 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 5 5 5 0 0 5 5 5 5 5 0 0 0 5 0 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 5 0 0 0
0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 5 5 5 5 5 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 5 5 5 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0

//Image 2

20 40 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ChainCode File:

20 40 0 1
1 2 8 6 5 4 5 7 0 7 5 4 4 6 6 6 6 6 6 6 7 7 7 0 7 6 1 1 1 0 0 7 0 7 7 1 1 0 0 7 7 2 1 0 2 1 7 7 7 1 1 1 1 0 1 1 4 4 4 4 4 3 1 0 1 1 0 0 3 3 3 3 3 3 3 5 5 5 5 5 5 5 0 0 7 7 0 7 5 4 4 4 4 4 7 7 5 5 4 4 3 2 2 2 3 3 2 3 3 2 2 6 6 5 5 5 5 5 3 3 2 2 2 2 2 2 2 4 3 2

-1 Boundary:

20 40 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0
0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
Decompressed File:

20 40 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0