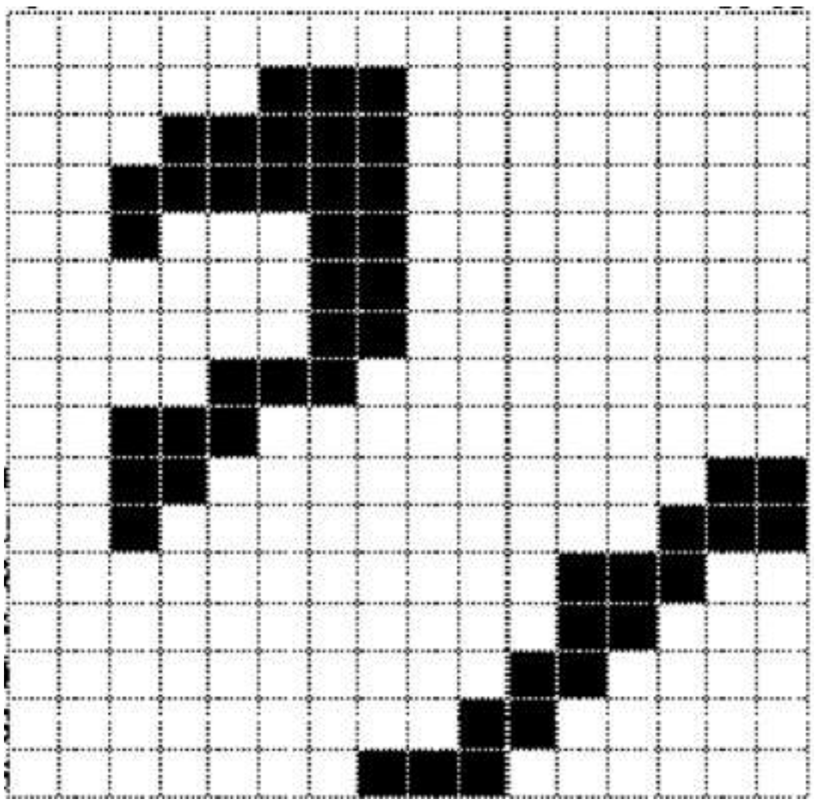
ASSIGNMENT 2: IMAGE COMPRESSION

Goal: The goal of this assignment is to get some practice with linked lists. In this assignment you will use linked lists (actually a list of lists) to encode an image in a compact format and perform operations on this format.

Problem Statement: Imagine a grayscale image of size NxN. Each pixel is either black (0) or white (1). In typical course, this will be represented as a 2D array storing O(N^2) values. The idea of the compressed representation is to use the redundancy in the pixel value information among neighboring pixels to reduce the amount of information that needs to be stored. The representation proceeds row-wise and stores the column indices of contiguous segments of black (0) pixels. For example, see the image below:



Its uncompressed representation is as follows (first line is image width and height):

16 16

1 1 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 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 0 1 1 1 0 0 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 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 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 0 0

1 1 0 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 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 0 0 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 0 0 0 1 1 1 1 1 1

Its equivalent compressed representation is:

16 16

-1

5 7 -1

3 7 -1

2 7 -1

2 2 6 7 -1

6 7 -1

6 7 -1

4 6 -1

2 4 -1

2 3 14 15 -1

2 2 13 15 -1

11 13 -1

11 12 -1

10 11 -1

9 10 -1

7 9 -1

The first line has image width and height respectively. Second line onwards, each line corresponds to image rows (we will number them as 0 to width-1). In each line we store column indices for contiguous segments of black pixels. For example in the line corresponding to row 2 the set of black pixels are from column number 3 to 7. Hence we have 3 7 in the fourth line of the representation. The -1s demarcate the rows. If there are more than 1 contiguous black segments in a row then we store the start and end indices of each segment in order, as for rows 4, 9 and 10.

Your job is to implement the following interface:

```
public interface compressedImage {

   public void compressedImage(String filename);

   public void compressedImage(boolean[][] grid, int width, int height);

   public boolean getPixelValue(int x, int y) throws pixelOutOfBoundException;

   public void setPixelValue(int x, int y, boolean val) throws pixelOutOfBoundException;

   public int[] numberOfBlackPixels();

   public void invert();

   public void performAnd(compressedImage img) throws boundsMismatchException;

   public void performOr(compressedImage img) throws boundsMismatchException;

   public void performXor(compressedImage img) throws boundsMismatchException;

   public String toStringUnCompressed();

   public String toStringCompressed();

}
```

Here, compressedImage constructors may either take in a 2D array of 0s and 1s, or a name of the file that has the same representation written in the textfile. getPixelValue and setPixelValue are operations to access and modify individual pixels. numberOfBlackPixels counts the number of black pixels in each row. Invert method makes each black pixel white and each white pixel black. Boolean operations over images are defined as: <u>for each pixel (i,j),</u> or returns white if any pixel is white, whereas and returns white if both pixels are white. Finally, xor returns white if exactly one of the pixels is white. It also has two methods to convert the current image into a string. The strings should be exactly the representation in either compressed or uncompressed form (including ~~line breaks~~ commas).

<u>Code:</u> Your code must compile and run on GCL machines.

<u>Desiderata:</u> Each operation in the input must be processed as efficiently as possible (in terms of time complexity).

<u>What is being provided?</u>

The folder contains three files :

1. CompressedImageInterface.java : This file contains the interface and the exceptions you will need to throw.

2. LinkedListImage.java : This file contains all the methods which you need to implement. The main method of this class contains tests for all the methods. Once you have implemented all the methods correctly, all the tests must be successful. You can refer to this link for understanding more about interfaces, if required:

https://docs.oracle.com/javase/tutorial/java/IandI/usinginterface.html

3. sampleInputFile.txt : This contains a sample input image in the uncompressed format.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run “unzip yourfile.zip” in addition to your code “writeup.txt” should be produced in the working directory.

   You will be penalized for any submissions that do not conform to this requirement.

2. The writeup.txt should have a line that lists names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone say None.
   After this line, you are welcome to write something about your code, though this is not necessary.

Evaluation Criteria

The assignment is worth 4 points. Your code will be autograded at the demo time against a series of tests. Two points will be provided for correct output and two points will be provided for your explanations of your code and your answering demo questions appropriately.

What is allowed? What is not?

1. This is an individual assignment.
2. Your code must be your own. You are not to take guidance from any general purpose code or problem specific code meant to solve these or related problems.
3. You are not allowed to use built-in (or anyone else’s) implementations of linked list data structures. A key aspect of the course is to have you learn how to implement these data structures.
4. You should develop your algorithm using your own efforts. You should not Google search for direct solutions to this assignment. However, you are welcome to Google search for generic Java-related syntax.
5. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class.** Please read academic integrity guidelines on the course home page and follow them carefully.
6. Your submitted code will be automatically evaluated against another set of benchmark problems. You get significant penalty if your output is not automatically parsable and does not follow input-guidelines.
7. We will run plagiarism detection software. Anyone found guilty will be awarded a suitable penalty as per IIT rules.