University of Science and Technology
School of Computational Sciences and Artificial Intelligence.
DSAI 325, Introduction to Information Theory

مدينة زويل للعلوم والتكنولوجيا
Zewail City of Science and Technology

Assignment 2: Implementation of LZ78

Spring 2025

Assignment Guidelines

- Each student does the assignment individually.
- The assignment will be graded out of 10 marks.
- Due date: Saturday, March 1 [11:59 PM].
- For every day of delay in delivery, 0.75 mark will be deducted.
- The assignment must be submitted through the course classroom before the due date. Then, assignments will be discussed and graded by your TA(s) in your Lab.
- The assignment submission must include the following:
    1. '.java' file(s) that contains the code.
    2. A report containing a copy of the code accompanied by two test cases. For each test case, the original text size and the compressed data size should be calculated.

Assignment Requirements

Students are required to implement Java code of the following methods:

- Compress method: This method will take as input a path of a txt file containing the plain text message to be compressed and return an output '.bin' file that contains the binary representation of the LZ78 tags as binary bytes.
- Decompress method: This method will take as input a path of a '.bin' file that contains the binary representation of the LZ78 tags to be decompressed and return an output txt file that contain the decompressed text.
- Main method: This method iteratively asks the user to enter his/her choice to compress, decompress, or terminate the program.

The Grading Criteria

		Marks
Code	Compress method	4
	Decompress method	4
	Main method, Code readability	1
Report		1

Kindly find below some programming tips:

- After obtaining the LZ78 tags, you should decide the number of bits needed to store the position part of the tag. You should find the maximum position number and find out how many bits needed for it.

```
// method to return the number of bits needed to code an integer number
public static int bitsNeeded(int number) {
	if (number == 0) {
		return 1; // Special case for 0
	}
	return (int)(Math.floor(Math.log(number) / Math.log(2))) + 1;
}

public static void main(String[] args) {
	// suppose that the max position number is equal to 9
	int maxPos = 9;
	System.out.println("The number " + maxPos + " needs " + bitsNeeded(maxPos) + " bits");
}
```

- You should initialize a binary string that will concatenate the binary representation of all the tags. For each tag, you should obtain its binary representation and append it to this binary string. The position part will be encoded as previously explained. The ASCII code of the next character will be encoded in 8 bits.

```
// take as input an integer number and #bits
// return the binary representation of the number of length #bits
public static String toBinary(int number, int bits) {
	String binaryString = Integer.toBinaryString(number);
	int length = binaryString.length();
	if (length < bits) {
		// Pad with leading zeros
		binaryString = String.format("%" + bits + "s", binaryString).replace(' ', '0');
	}
	return binaryString;
}

public static String BinarizeTag(int pos, int bits, char nextChar) {
	// the position will be encoded in certain number of bits
	String binaryPos = toBinary(pos, bits);

	// the nextChar will be encoded in exactly 8 bits
	int asciiCode = (int) nextChar;
	String binaryChar = toBinary(asciiCode, 8);

	// return the string concatenation of binary representation of pos and nextChar
	return binaryPos+binaryChar;
}

public static void main(String[] args) {
	// the binary string to hold all the tags
```

```
        String binary_str = "";

        // suppose that i have this tag <7,'A'> to be added to the binary String
        binary_str = binary_str + BinarizeTag(7, 4, 'A');

        System.out.println("After adding The tag <7,\'A\' to the binary string");
        System.out.println("The binary string = "+binary_str);

        // suppose that i have this tag <0,'B'> to be added to the binary String
        binary_str = binary_str + BinarizeTag(0, 4, 'B');

        System.out.println("After adding The tag <0,\'B\' to the binary string");
        System.out.println("The binary string = "+binary_str);
}
```

The previous code output:
After adding The tag <7,'A' to the binary string
The binary string = 011101000001
After adding The tag <0,'B' to the binary string
The binary string = 011101000001000001000010

- The below code can be utilized to write the tags as bytes to the binary file:

```
import java.io.FileOutputStream;
import java.io.IOException;

public class Main
{
        public static void main(String[] args) {
                // suppose that the tags we have are <7,'A'>,<0,'B'>
                String binaryString = "011101000001000001000010";

                // create byte array of suitable length
                byte[] byteArray = new byte[binaryString.length() / 8];

                for (int i = 0; i < byteArray.length; i++) {
                        // segment the binary string into bytes (each 8 bit)
                        String byteString = binaryString.substring(8 * i, 8 * (i + 1));
                        byteArray[i] = (byte) Integer.parseInt(byteString, 2);
                }

                // write the bytes array to the output binary file

                try (FileOutputStream fos = new FileOutputStream("binaryfile.bin")) {
                        fos.write(byteArray);
                        System.out.println("Binary data written to file.");
                } catch (IOException e) {
                        e.printStackTrace();
```

```
            }

        }
}
```

- While writing the binary representation of the tags, you actually write bytes not bits. So, you may need to append some zero bits to the end of the binary string. In addition, this issue needs to be handled in the reading the tags in the decompression method.

- The below code can be utilized to read the tags as bytes and form the binary string that contain the tags:

```
import java.io.FileInputStream;
import java.io.IOException;

public class Main
{
        public static void main(String[] args)
        {

                try (FileInputStream fis = new FileInputStream("binaryfile.bin"))
                {

                        byte[] byteArray2 = fis.readAllBytes();
                        StringBuilder binaryString2 = new StringBuilder();

                        for (byte b : byteArray2)
                        {
                                String byteString = String.format("%8s", Integer.toBinaryString(b &
                                0xFF)).replace(' ', '0');
                                binaryString2.append(byteString);
                        }

                        System.out.println("Binary data read from file: " + binaryString2.toString());
                }
                catch (IOException e)
                {
                        e.printStackTrace();
                }

        }
}
        // The previous code will exactly reconstruct the tags binary string and output the following:
        Binary data read from file: 011101000001000001000010
```

- In the decompression part, you will get all the tags as a single binary string. You should be able to segment each tag in separate. In addition, you should segment the position from the next character. So, you should find a suitable way to inform the decompression method how many bits are used for encoding the position part of each tag.