

Problem 1

```
/**  
* CT255 - Assignment 4  
* Skeleton code for Steganography assignment.  
*  
* @author Michael Schukat  
* @version 1.0  
*/  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
// binary for ciaran: 011000110110100101100001011100100110000101101110  
//binary for gray: 01100111 01110010 01100001 01111001  
public class Steganol {  
    /**  
     * Constructor for objects of class Steganol  
     */  
    public Steganol() {  
    }  
  
    public static void main(String[] args) {  
        String arg1, arg2, arg3, arg4;  
        boolean err = false;  
        if (args != null && args.length > 1) { // Check for minimum number  
of arguments  
            arg1 = args[0];  
            arg2 = args[1];  
            if (arg2.isEmpty()) {  
                err = true;  
            } else if ((arg1.equals("A")) && (args.length > 3)) {  
                // Get other arguments  
                arg3 = args[2];  
                arg4 = args[3];  
                if ((arg3.isEmpty()) || (arg4.isEmpty())) {  
                    err = true;  
                } else {  
                    // Hide bitstring  
                    hide(arg2, arg3, arg4);  
                }  
            } else if (arg1.equals("E")) {  
                // Extract bitstring from text  
                retrieve(arg2);  
            } else {  
                err = true;  
            }  
        } else {  
            err = true;  
        }  
        if (err) {  
            System.out.println();  
            System.out.println("Use: Steganol <A:E><Input  
File><OutputFile><Bitstring>");  
            System.out.println("Example: Steganol A inp.txt out.txt  
0010101");  
            System.out.println("Example: Steganol E inp.txt");  
        }  
    }  
  
    static void hide(String inpFile, String outFile, String binString) {
```

```

BufferedReader reader;
BufferedWriter writer;
try {
    reader = new BufferedReader(new FileReader(inpFile));
    writer = new BufferedWriter(new FileWriter(outFile));
    String line = reader.readLine();
    int bitIndex = 0;

    while (line != null) {
        // store amended line in output file
        if (bitIndex < binString.length()) {
            //problem 1
            //save the current index of bit string as a character
            char bit = binString.charAt(bitIndex);

            if (bit == '0') {
                line += " "; // add one space to the line for 0
            } else if (bit == '1') {
                line += "  "; // add two spaces to the line for 1
            }
            bitIndex++; //increase the index of the bit string
        }
        writer.write(line);
        writer.newLine();
        // read next line
        line = reader.readLine();
    }
    reader.close();
    writer.close();
} catch (IOException e) {
    e.printStackTrace();
}
}

static void retrieve(String inpFile) {
    BufferedReader reader;
    try {
        reader = new BufferedReader(new FileReader(inpFile));
        String line = reader.readLine();

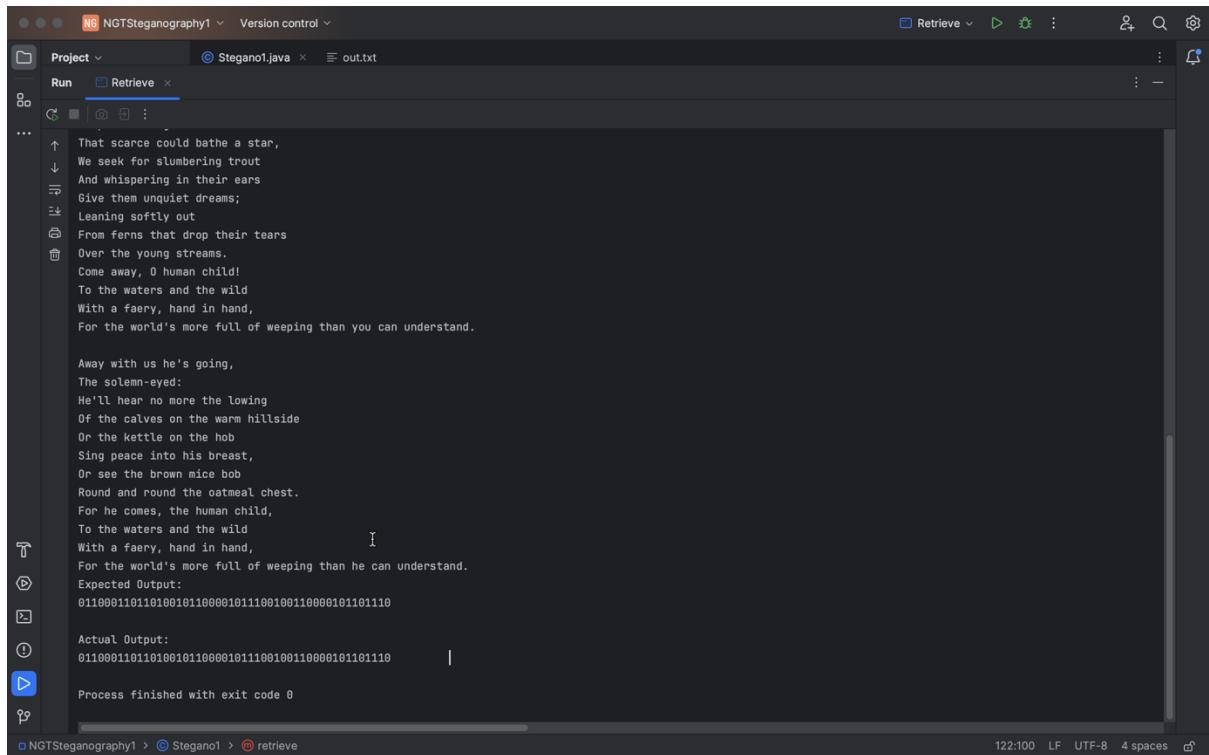
        String result = ""; // string to hold the binary result
        while (line != null) {
            // Your code starts here
            System.out.println(line); //read the next line
            // if the line ends with two spaces, add 1 to result
            if (line.endsWith("  ")) {
                result += "1";
            }
            // if the line ends with one space, add 0 to result
            else if (line.endsWith(" ")) {
                result += "0";
            }
            //if the line does not end with a space, add nothing to
result
            else {
                result += " ";
            }
            line = reader.readLine();
        }
        System.out.println("Expected Output:
\n011000110110100101110010011000010110110\n");
        System.out.println("Actual Output: \n" + result);
        reader.close();
    }
}

```

```

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

```



```

That scarce could bathe a star,
We seek for slumbering trout
And whispering in their ears
Give them unquiet dreams;
Leaning softly out
From ferns that drop their tears
Over the young streams.
Come away, O human child!
To the waters and the wild
With a faery, hand in hand,
For the world's more full of weeping than you can understand.

Away with us he's going,
The solemn-eyed:
He'll hear no more the lowing
Of the calves on the warm hillside
Or the kettle on the hob
Sing peace into his breast,
Or see the brown mice bob
Round and round the oatmeal chest.
For he comes, the human child,
To the waters and the wild
With a faery, hand in hand,
For the world's more full of weeping than he can understand.

Expected Output:
011000011011010010110000101110010011000010110110

Actual Output:
011000011011010010110000101110010011000010110110

Process finished with exit code 0

```

Problem 2

```

/**
* CT255 - Assignment 4
* Skeleton code for Steganography assignment.
*
* @author Michael Schukat
* @version 1.0
*/
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

// binary for ciaran: 011000011011010010110000101110010011000010110110
// binary for gray: 01100111011100100110000101111001
public class Steganol {
    /**
     * Constructor for objects of class Steganol
     */
    public Steganol() {
    }

    public static void main(String[] args) {
        String arg1, arg2, arg3, arg4;
        boolean err = false;
    }
}

```

```

        if (args != null && args.length > 1) { // Check for minimum number
of arguments
            arg1 = args[0];
            arg2 = args[1];
            if (arg2.isEmpty()) {
                err = true;
            } else if ((arg1.equals("A")) && (args.length > 3)) {
                // Get other arguments
                arg3 = args[2];
                arg4 = args[3];
                if ((arg3.isEmpty()) || (arg4.isEmpty())) {
                    err = true;
                } else {
                    // Hide bitstring
                    hide(arg2, arg3, arg4);
                }
            } else if (arg1.equals("E")) {
                // Extract bitstring from text
                retrieve(arg2);
            } else {
                err = true;
            }
        } else {
            err = true;
        }
        if (err) {
            System.out.println();
            System.out.println("Use: Steganol <A:E><Input
File><OutputFile><Bitstring>");
            System.out.println("Example: Steganol A inp.txt out.txt
0010101");
            System.out.println("Example: Steganol E inp.txt");
        }
    }

    static void hide(String inFile, String outFile, String binString) {
        BufferedReader reader;
        BufferedWriter writer;
        try {
            reader = new BufferedReader(new FileReader(inFile));
            writer = new BufferedWriter(new FileWriter(outFile));
            String line = reader.readLine();
            int bitIndex = 0;

            //padding bit if the length is odd
            if (binString.length() % 2 != 0) {
                binString += "0";
            }

            while (line != null) {
                if (bitIndex < binString.length()) {
                    //Problem 2
                    //divides the bitstring into a substring of 2 length,
and checks that substring for bit values
                    int bitIndextwo = bitIndex + 2;
                    String twobits = binString.substring(bitIndex,
bitIndextwo); //substring
                    //adds a certain amount of spaces to the end of the
lines depending on what digits are in the substring
                    if (twobits.equals("00")) {
                        line += " "; // 1 space for 00
                    }
                    else if (twobits.equals("01")) {

```

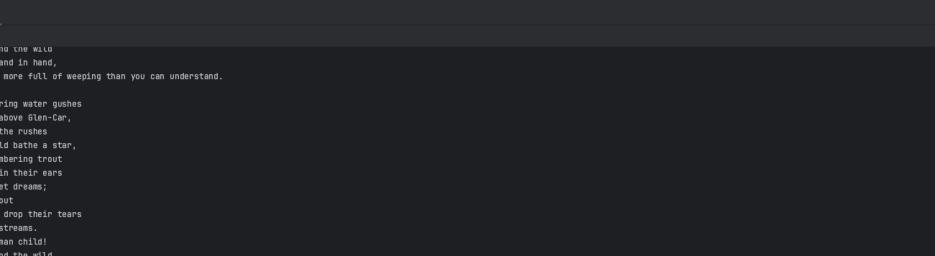
```

        line += "  "; // 2 for 01
    }
    else if (twobits.equals("10")) {
        line += "  "; // 3 for 10
    }
    else if (twobits.equals("11")) {
        line += "  "; // 4 for 11
    }
}
writer.write(line);
writer.newLine();
// read next line
line = reader.readLine();
bitIndex += 2; //increase the index of the bitstring
}
reader.close();
writer.close();
} catch (IOException e) {
    e.printStackTrace();
}
}

static void retrieve(String inpFile) {
    BufferedReader reader;
    try {
        reader = new BufferedReader(new FileReader(inpFile));
        String line = reader.readLine();
        String result = ""; // string to hold the binary result
        while (line != null) {
            // Your code starts here
            System.out.println(line); //read the next line
            //Problem 2
            //checks firstly for 4 spaces, then 3, then 2, then 1.
            //adds back the corresponding binary digits to the number
            of spaces

            if (line.endsWith("    ")) {
                result += "11";
            } else if (line.endsWith("   ")) {
                result += "10";
            } else if (line.endsWith("  ")) {
                result += "01";
            } else if (line.endsWith(" ")) {
                result += "00";
            }
            else {
                result += "  "; //if theres no spaces, it adds nothing
            }
            line = reader.readLine();
        }
        System.out.println("Expected Output:
\n01100111011100100110000101111001\n");
        System.out.println("Actual Output: \n" + result);
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```



```
if (line.endsWith(" ")) {  
    out.println(line);  
}  
  
Where the wandering water gushes  
From the hills above Glen-Car,  
In pools among the rushes  
That scarce could bathe a star,  
We seek for slumbering trout  
And whispering in their ears  
Give them unquiet dreams;  
Leaning softly out  
From ferns that drop their tears  
Over the young streams.  
Come away, O human child!  
To the waters and the wild  
With a faery, hand in hand,  
For the world's more full of weeping than you can understand.  
  
Away with us he's going,  
The solemn-eyed:  
He'll hear no more the lowing  
Of the calves on the warm hillside  
Or the kettle on the hob  
Singing merrily to itself,  
Or see the brown mice bob  
Round and round the oatmeal chest.  
For he consaps, the human child,  
To the waters and the wild  
With a faery, hand in hand,  
For the world's more full of weeping than he can understand.  
Expected Output:  
011001110110010011000001011111001  
  
Actual Output:  
011001110110010011000001011111001  
  
Process finished with exit code 0
```