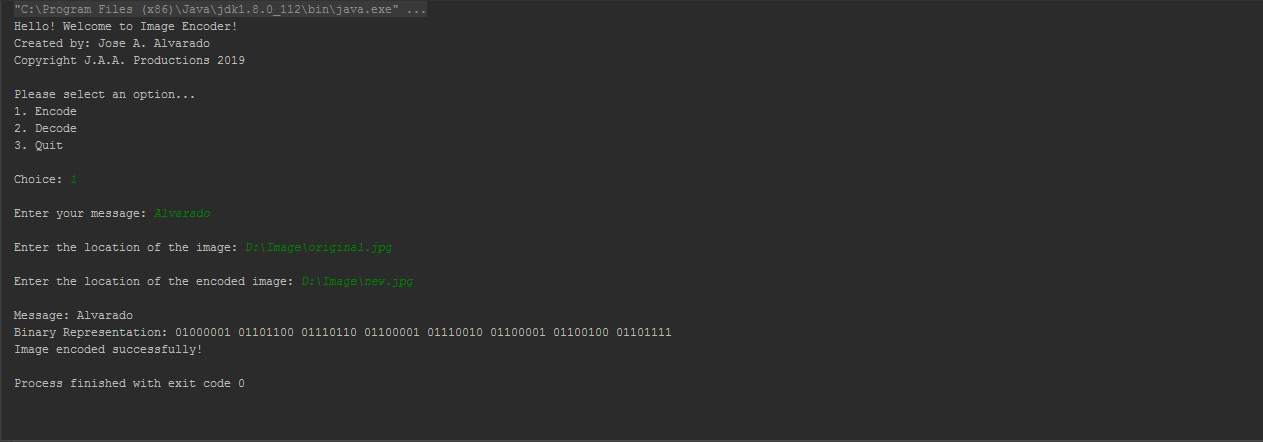
**Homework #2**

1. Find the Binary representation of “your last name” from ASCII table.



1. Identify the Binary representation of the following picture:



*Sample Output (Partial):*

Image in Binary...  
Pixel (x:0, y:0):   
Current Letter: A - [0]1000001  
11111111001001010010110001011000 -> 11111111001001010010110001011000  
  
Pixel (x:1, y:0):   
Current Letter: A - 0[1]000001  
11111111001001010010110001011000 -> 11111111001001010010110001011001  
  
Pixel (x:2, y:0):   
Current Letter: A - 01[0]00001  
11111111001001000010101101010111 -> 11111111001001000010101101010110  
  
Pixel (x:3, y:0):   
Current Letter: A - 010[0]0001  
11111111001001000010101101010111 -> 11111111001001000010101101010110  
  
Pixel (x:4, y:0):   
Current Letter: A - 0100[0]001  
11111111001000110010101001010110 -> 11111111001000110010101001010110  
  
Pixel (x:5, y:0):   
Current Letter: A - 01000[0]01  
11111111001000110010101001010110 -> 11111111001000110010101001010110  
  
Pixel (x:6, y:0):   
Current Letter: A - 010000[0]1  
11111111001000110010101001010110 -> 11111111001000110010101001010110  
  
Pixel (x:7, y:0):   
Current Letter: A - 0100000[1]  
11111111001000100010100101010101 -> 11111111001000100010100101010101  
  
Pixel (x:8, y:0):   
Current Letter: l - [0]1101100  
11111111001000110010101001010110 -> 11111111001000110010101001010110  
  
Pixel (x:9, y:0):   
Current Letter: l - 0[1]101100  
11111111001000110010101001010110 -> 11111111001000110010101001010111  
  
Pixel (x:10, y:0):   
Current Letter: l - 01[1]01100  
11111111001000100010100101010101 -> 11111111001000100010100101010101  
  
Pixel (x:11, y:0):   
Current Letter: l - 011[0]1100  
11111111001000100010100101010101 -> 11111111001000100010100101010100  
  
Pixel (x:12, y:0):   
Current Letter: l - 0110[1]100  
11111111001000110010101001010110 -> 11111111001000110010101001010111  
  
Pixel (x:13, y:0):   
Current Letter: l - 01101[1]00  
11111111001001000010101101010111 -> 11111111001001000010101101010111  
  
Pixel (x:14, y:0):   
Current Letter: l - 011011[0]0  
11111111001001010010110001011000 -> 11111111001001010010110001011000  
  
Pixel (x:15, y:0):   
Current Letter: l - 0110110[0]  
11111111001001100010110101011001 -> 11111111001001100010110101011000  
  
Pixel (x:16, y:0):   
Current Letter: v - [0]1110110  
11111111001001110010111001011100 -> 11111111001001110010111001011100  
  
Pixel (x:17, y:0):   
Current Letter: v - 0[1]110110  
11111111001010000010111101011101 -> 11111111001010000010111101011101

1. Create a new picture by embedding the Binary representation of your last name in above picture’s binary representation by using the LSB method.

Original





*Encoded Image*

1. Prepare a program to discover the word of your last name from the new picture.

*The following is the Kotlin source code:*

/\*  
\* Created By: Jose A. Alvarado  
\* \*/  
  
import java.awt.image.BufferedImage  
import java.io.File  
import java.io.FileOutputStream  
import java.lang.StringBuilder  
import javax.imageio.ImageIO  
  
class Steganography {  
 companion object {  
 private const val delimiter = "11111110"  
  
 private fun convertMessageToBinary(name : String) : Pair<ArrayList<String>, Map<String, Char>> {  
 val binaryNameArrayList = *arrayListOf*<String>()  
 val charBinMap = *mutableMapOf*<String, Char>()  
  
 *println*("Message: $name")  
 *print*("Binary Representation: ")  
  
 name.*forEach* **{** val currCharBinRepresentation = String.*format*("%08d", Integer.toBinaryString(**it**.toInt()).*toInt*())  
 *print*("$currCharBinRepresentation ")  
 charBinMap[currCharBinRepresentation] = **it** binaryNameArrayList.add(currCharBinRepresentation)  
 **}** *println*()  
  
 return Pair(binaryNameArrayList, charBinMap)  
 }  
  
 fun generateEncodedImage(image : BufferedImage, picturePath : String) = ImageIO.write(image, "png", File(picturePath))  
  
 fun embedMessage(picture : BufferedImage, name : String) : BufferedImage {  
 val originalBufferedImg : BufferedImage = picture  
 val picWidth : Int = originalBufferedImg.*width* val picHeight : Int = originalBufferedImg.*height* val newBufferedImage = BufferedImage(picWidth, picHeight, originalBufferedImg.*type*)  
  
 val convertMessageToBinaryResult = convertMessageToBinary(name)  
 val binConvertedNameArrayList = convertMessageToBinaryResult.first  
 val binConvertedNameCharBinMap = convertMessageToBinaryResult.second  
  
 val separator = System.lineSeparator()  
 val file = File(".\\debug.txt")  
 val outputStream = FileOutputStream(file)  
  
 outputStream.write("Message: $name$separator$separator".*toByteArray*())  
 outputStream.write("Binary Representation:$separator".*toByteArray*())  
  
 val binConvertedNameStringBuilder = StringBuilder("")  
 binConvertedNameArrayList.*forEach* **{** binConvertedNameStringBuilder.append(**it**)  
 outputStream.write("$**it** ".*toByteArray*())  
 **}** binConvertedNameStringBuilder.append(delimiter)  
  
 val binConvertedNameCharArray = binConvertedNameStringBuilder.toString().*toCharArray*()  
  
 outputStream.write("$separator$separator".*toByteArray*())  
 outputStream.write("Image in Binary...$separator".*toByteArray*())  
  
 var count = 0  
 for (yIndex in (0 *until* picHeight)) {  
 for (xIndex in (0 *until* picWidth)) {  
 val pixel = originalBufferedImg.getRGB(xIndex, yIndex)  
 val rgbBinStr = Integer.toBinaryString(pixel)  
 outputStream.write("Pixel (x:$xIndex, y:$yIndex): $separator".*toByteArray*())  
  
 if (count < binConvertedNameCharArray.size) {  
 val currLetterIndex = count / 8  
 val currBitIndex = count % 8  
  
 if (currLetterIndex < binConvertedNameArrayList.size) {  
 val currLetterBin = binConvertedNameArrayList[currLetterIndex]  
 val currLetter = binConvertedNameCharBinMap[currLetterBin]  
  
 val currLetterBinStrBuilder = StringBuilder(currLetterBin)  
 val tempChar = currLetterBinStrBuilder[currBitIndex]  
 currLetterBinStrBuilder.replace(currBitIndex, currBitIndex + 1, "[")  
 currLetterBinStrBuilder.insert(currBitIndex + 1, "$tempChar]")  
  
  
 outputStream.write("Current Letter: ${if (currLetter!!.*isWhitespace*()) "[SPACE]" else currLetter} - $currLetterBinStrBuilder$separator".*toByteArray*())  
 }  
 else {  
 val delimiterStrBuilder = StringBuilder(delimiter)  
 val tempChar = delimiterStrBuilder[currBitIndex]  
 delimiterStrBuilder.replace(currBitIndex, currBitIndex + 1, "[")  
 delimiterStrBuilder.insert(currBitIndex + 1, "$tempChar]")  
  
 outputStream.write("Current Letter: [DELIMITER] - $delimiterStrBuilder$separator".*toByteArray*())  
 }  
  
 val rgbBinCharArray = rgbBinStr.*toCharArray*()  
 rgbBinCharArray[rgbBinCharArray.*lastIndex*] = binConvertedNameCharArray[count]  
  
 val newRgbBinStrBuilder = StringBuilder("")  
 rgbBinCharArray.*forEach* **{** newRgbBinStrBuilder.append(**it**) **}** val newRgbBinStr = newRgbBinStrBuilder.toString()  
 val newPixel = Integer.parseUnsignedInt(newRgbBinStr, 2)  
  
 outputStream.write("$rgbBinStr -> $newRgbBinStrBuilder$separator".*toByteArray*())  
  
 newBufferedImage.setRGB(xIndex, yIndex, newPixel)  
 count++  
 }  
 else {  
 newBufferedImage.setRGB(xIndex, yIndex, pixel)  
 outputStream.write("NO ENCODING$separator$rgbBinStr$separator".*toByteArray*())  
 }  
  
 outputStream.write(separator.*toByteArray*())  
 }  
 }  
  
 outputStream.flush()  
 outputStream.close()  
  
 return newBufferedImage  
 }  
  
 fun retrieveEncodedMessageFromImage(picturePath: String) : String {  
 val bufferedImg : BufferedImage = ImageIO.read(File(picturePath))  
 val picWidth : Int = bufferedImg.*width* val picHeight : Int = bufferedImg.*height* val tempChar = StringBuilder("")  
 val encodedMessageStringBuilder = StringBuilder("")  
  
 var count = 0  
 outer@ for ((\_, yIndex) in (0 *until* picHeight).*withIndex*()) {  
 for ((\_, xIndex) in (0 *until* picWidth).*withIndex*()) {  
 val pixel = bufferedImg.getRGB(xIndex, yIndex)  
 val rgbBinStr = Integer.toBinaryString(pixel)  
  
 val rgbBinCharArray = rgbBinStr.*toCharArray*()  
  
 if (count < 8) {  
 tempChar.append(rgbBinCharArray[rgbBinCharArray.*lastIndex*])  
 count++  
  
 if (count == 8) {  
 count = 0  
  
 if (tempChar.toString() == delimiter) {  
 tempChar.*clear*()  
 break@outer  
 }  
 else {  
 encodedMessageStringBuilder.append(Integer.parseInt(tempChar.toString(), 2).toChar())  
 tempChar.*clear*()  
 }  
 }  
 }  
 }  
 }  
  
 return encodedMessageStringBuilder.toString()  
 }  
 }  
}

import Steganography.Companion.embedMessage  
import Steganography.Companion.generateEncodedImage  
import Steganography.Companion.retrieveEncodedMessageFromImage  
import java.io.File  
import javax.imageio.ImageIO  
  
fun main() {  
  
 *println*("Hello! Welcome to Image Encoder!")  
 *println*("Created by: Jose A. Alvarado")  
 *println*("Copyright J.A.A. Productions 2019")  
 *println*()  
 *println*("Please select an option...")  
 *println*("1. Encode")  
 *println*("2. Decode")  
 *println*("3. Quit")  
 *println*()  
 *print*("Choice: ")  
  
 val userChoice = *readLine*()!!  
  
 when (Integer.parseInt(userChoice)) {  
 1 -> {  
 *println*()  
 *encode*()  
 }  
 2->{  
 *println*()  
 *decode*()  
 }  
 3-> {  
 *println*("Goodbye!")  
 return  
 }  
 else -> *println*("Invalid Choice")  
 }  
}  
  
fun encode() {  
 *print*("Enter your message: ")  
 val nameInput = *readLine*()!!  
 *println*()  
 *print*("Enter the location of the image: ")  
 val originalPath = *readLine*()!!  
 *println*()  
 *print*("Enter the location of the encoded image: ")  
 val newPath = *readLine*()!!  
 *println*()  
  
 generateEncodedImage(embedMessage(ImageIO.read(File(originalPath)), nameInput), newPath)  
 *println*("Image encoded successfully!")  
}  
  
fun decode() {  
 *print*("Enter the location of the encoded image: ")  
 val picPath = *readLine*()!!  
  
 *println*("The message is \"${retrieveEncodedMessageFromImage(picPath)}\"")  
}

**Please let me know if you have any questions and if you would like me to run my code for you.**