NAME- SWARNAVA BOSE

ROLLNO- 24CS06014

ASSIGNMENT 10

1. Develop a program to embed a short text message within an image using the Least Significant Bit (LSB) technique. Assignment Details:

● Write a program that hides a short text message within the LSBs of an image file (e.g., .bmp or .png).

● Allow the user to specify both the image file and the message to be embedded.

● Implement a function to retrieve and display the hidden message from the image.

● Provide an option for the user to choose between embedding the message in the red, green, or blue colour channels.

● Only short text messages should be embedded, with no encryption applied.

● Assume the message embedding starts from pixel (1,1).

Answer:-

```
#include <opencv2/opencv.hpp>

#include <iostream>

#include <bitset>


using namespace cv;

using namespace std;
```

```
// Embed message in the LSB of the specified color channel
void embedMessage(Mat &image, const string &message, const string &channel) {
    int colorChannel;
    if (channel == "red") colorChannel = 2;
    else if (channel == "green") colorChannel = 1;
    else if (channel == "blue") colorChannel = 0;
    else {
        cout << "Invalid color channel" << endl;
        return;
    }

    string messageBits = "";
    for (char c : message) {
        messageBits += bitset<8>(c).to_string();
    }
    messageBits += "00000000"; // Null character to signify end of message

    int idx = 0;
    for (int y = 0; y < image.rows && idx < messageBits.size(); ++y) {
        for (int x = 0; x < image.cols && idx < messageBits.size(); ++x) {
            Vec3b &pixel = image.at<Vec3b>(y, x);
            pixel[colorChannel] = (pixel[colorChannel] & ~1) | (messageBits[idx] - '0');
```

```
            idx++;
        }
    }
}

// Retrieve message from the LSB of the specified color channel
string retrieveMessage(const Mat &image, const string &channel) {
    int colorChannel;
    if (channel == "red") colorChannel = 2;
    else if (channel == "green") colorChannel = 1;
    else if (channel == "blue") colorChannel = 0;
    else {
        cout << "Invalid color channel" << endl;
        return "";
    }

    string messageBits = "";
    for (int y = 0; y < image.rows; ++y) {
        for (int x = 0; x < image.cols; ++x) {
            const Vec3b &pixel = image.at<Vec3b>(y, x);
            messageBits += to_string(pixel[colorChannel] & 1);

            // Check for the null character (8 bits of 0)
```

```
            if (messageBits.size() >= 8 &&
                messageBits.substr(messageBits.size() - 8) == "00000000") {
                messageBits = messageBits.substr(0, messageBits.size() - 8);
                string message = "";
                for (size_t i = 0; i < messageBits.size(); i += 8) {
                    bitset<8> charBits(messageBits.substr(i, 8));
                    message += char(charBits.to_ulong());
                }
                return message;
            }
        }
    }
    return "";
}

int main() {
    string imagePath, outputImagePath, message, channel;
    cout << "Enter the image file path: ";
    cin >> imagePath;
    cout << "Enter the message to embed: ";
    cin.ignore();
    getline(cin, message);
    cout << "Enter the output image file path: ";
```

```
    cin >> outputImagePath;
    cout << "Choose color channel (red, green, blue): ";
    cin >> channel;

    Mat image = imread(imagePath);
    if (image.empty()) {
        cerr << "Could not open or find the image." << endl;
        return -1;
    }

    embedMessage(image, message, channel);
    imwrite(outputImagePath, image);
    cout << "Message embedded successfully." << endl;

    // Retrieve the message to confirm
    Mat outputImage = imread(outputImagePath);
    string extractedMessage = retrieveMessage(outputImage, channel);
    cout << "Retrieved Message: " << extractedMessage << endl;

    return 0;
}
```

- **embedMessage**: This function embeds the message in the LSB of the specified color channel by setting the last bit of each pixel channel value.

- **retrieveMessage**: This function extracts the bits from the specified color channel's LSBs, reconstructs the characters, and stops when it detects a null character (indicating the end of the message).
- **Main Function**: Takes inputs for the image path, message, output path, and channel, embeds the message, saves the new image, and then retrieves the message for confirmation.

To run this code, install OpenCV:

```
sudo apt-get install libopencv-dev

g++ -o lsb_steganography lsb_steganography.cpp `pkg-config --cflags --libs opencv4`
```

After compiling, run:

```
./lsb_steganography
```