1. White (R: 255, G: 255, B: 255), Silver (R: 192, G: 192, B: 192), Coral (R: 255, G: 127, B: 80)
2. The lower the red values were, the darker it was.
3. Divide the binary number by 4 and then multiply it by 4
4. 183 starts off as 1011 0111, when it gets divided by 4 or 0100, the number becomes 0010 1101. Then when you multiply it by for or 0100, the number becomes 1011 0100 which is 183 with the last 2 numbers set as 0.
5. Table below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 0000 0101 | 1 | 0000 0001 | 4 | 0000 0100 |
| 80 | 0101 0000 | 17 | 0001 0100 | 80 | 0101 0000 |

1. No, because the difference is too little for the human eye to easily detect.
2. Divide by 64
3. Isolate the right most 2 bits with “%4” and then multiply it by 64 to expand it from the range 0 to 256.
4. I would divide it by 10, 100, or whatever the place is to make sure the number we want to isolate is at the right-most side. Then I would do “%10” to the number to completely isolate the right-most number.
5. You don’t need to resize arch.jpg for it to fit in beach.jpg but you will need to resize beach.jpg to fit in arch.jpg.
6. Beach.jpg would need to become less wide for it to fit in arch.jpg.
7. R: 232, G: 174, B: 92
8. Light orange
9. The lower 2 bits are replaced because they barely affect the color of the source picture
10. The highest 2 bits are used to reveal secret.jpg because they greatly affect the color of the pixels
11. It almost looks pixelated because the process of hiding and revealing is not perfect and can cause small changes from the original picture causing it to look different
12. Instead of just replacing the 1st 2 bits with the last 2 bits of the secret image, the algorithm can replace even more bits (ex. 3 or 4). This means that the revealed image will look more like the original secret image because the colors are more accurate. But this also means that the combined image will show greater hints of the secret image because changing more bits means more change in color.
13. Int row = (int)(Math.random() \* source.getPixel2D().length) - height;
14. Int column = (int)(Math.random() \* source.getPixel2D()[0].length) - width;
15. Yes, the random is within bounds of the source. But the secret must be within the size of the source.
16. 2 bits: 4, 4 bits: 16, 8 bits: 256