```
Algorithm MethodOne

Input: Image: tensor $n \times n \times 3$
Output: NewImage: tensor $n \times n \times 3$
channel ← 1;
for every channel do
    #DCTimage is a matrix of dimension $n \times n$
    DCTimage ← calculateDCT(Image(:, :, channel));
    for row,col in DCTimage do
        with probability 0.5 do
            DCTimage(row, col) = 0; #except DCTimage(1,1) that cannot be reset
        end
    end
    #inverse of the perturbated image
    NewImage(:, :, channel) <- inverseDCT(DCTimage); end
```

```
Algorithm MethodTwo

Input: Image: tensor $n \times n \times 3$
Output: NewImage: tensor $n \times n \times 3$
channel ← 1;
for every channel do
    #DCTimage is a matrix of dimension $n \times n$
    DCTimage ← calculateDCT(Image(:, :, channel));
    Sigma = standardDeviation(Image)/2;
    for row,col in DCTimage do
        DCTimage(row, col) += sigma · random number $z \sim U\left(-\frac{1}{2}, \frac{1}{2}\right)$;
        # except DCTimage(1,1) that cannot be modified
    end
    #inverse of the perturbated image
    NewImage(:, :, channel) <- inverseDCT(DCTimage);
end
```