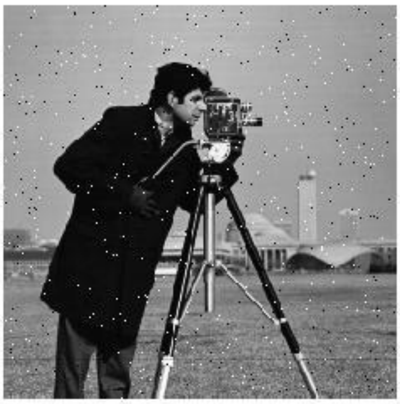
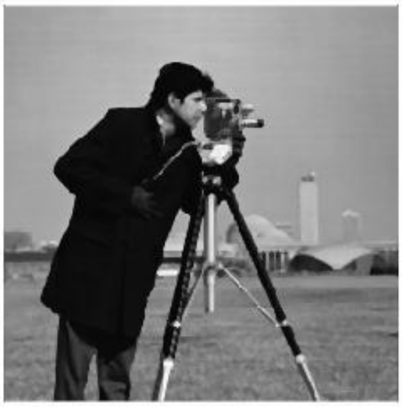
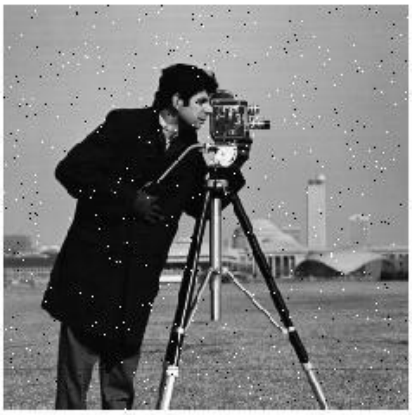
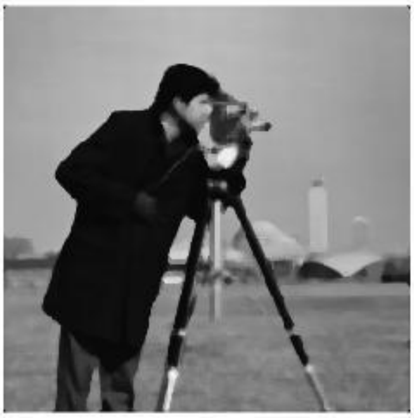
1) Implemented median filter by adding each coordinate and surrounding pixels in a matrix then take the median value of the matrix and assign it to the new image.

Cameraman1 with 3, 5, and 7 median filters:



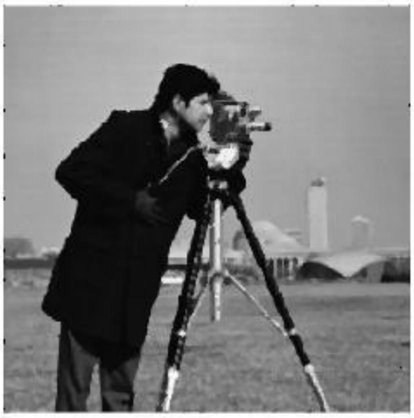


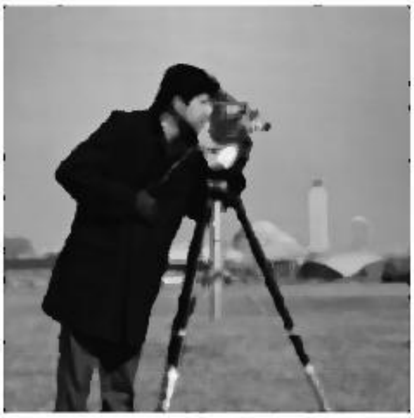


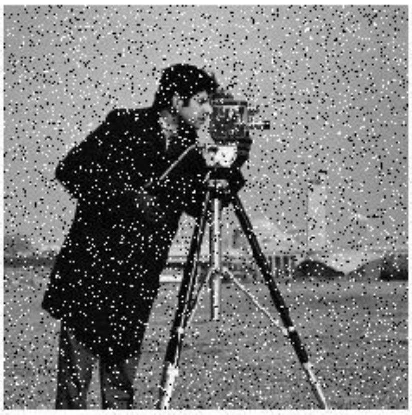


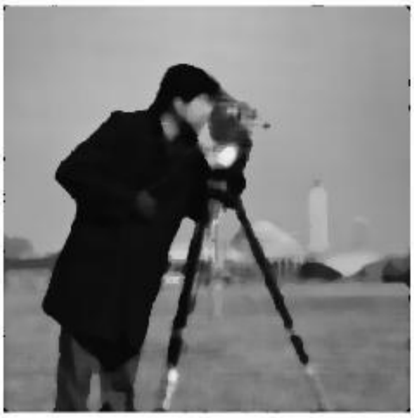
Cameraman2 with 3, 5, and 7 median filters:

Cameraman3 with 3, 5, and 7 median filters:

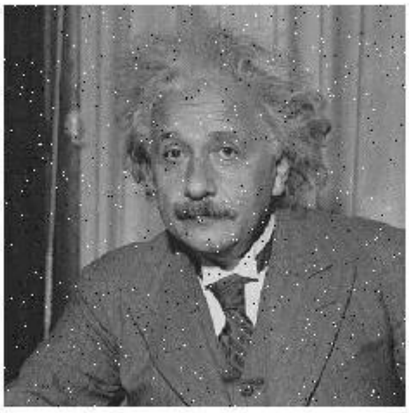


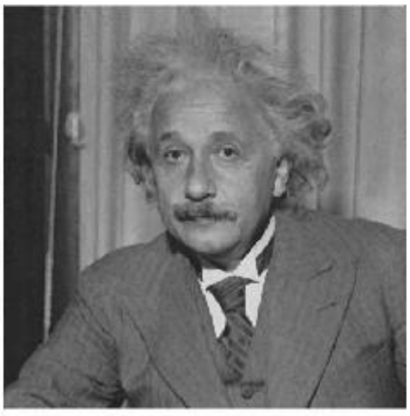


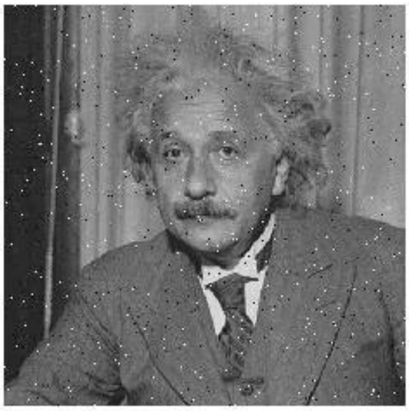
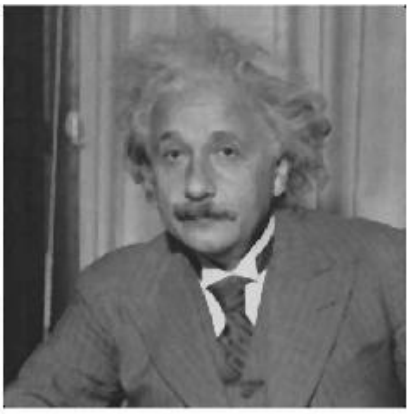
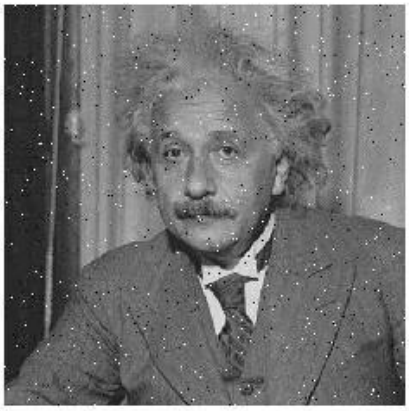
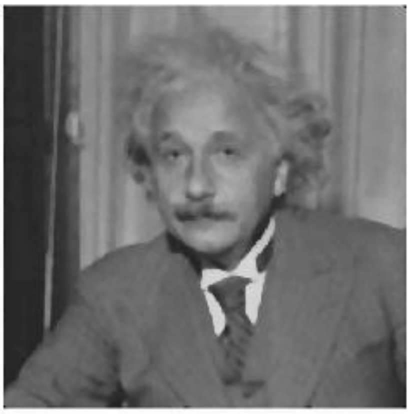




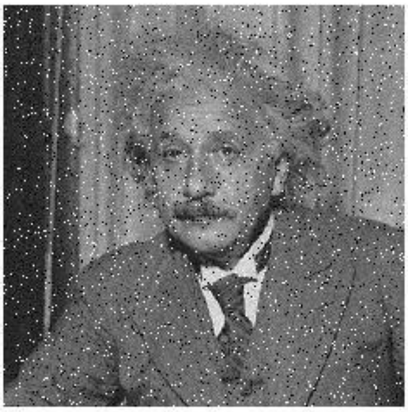
einstein1 with 3, 5 and 7 median filters:

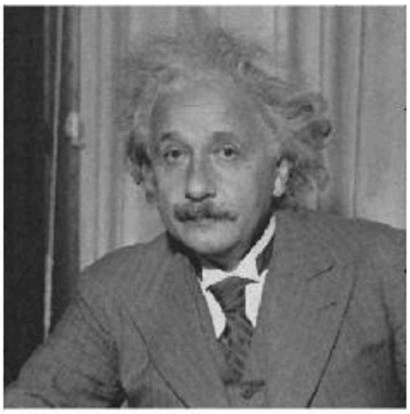


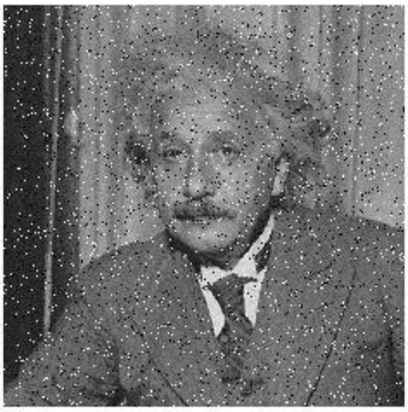


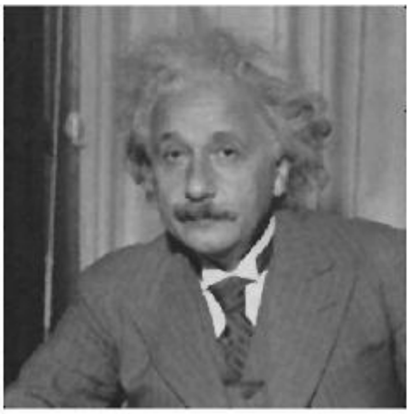





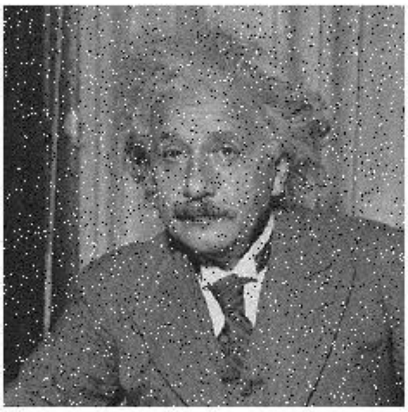
einstein2 with 3, 5 and 7 median filters:

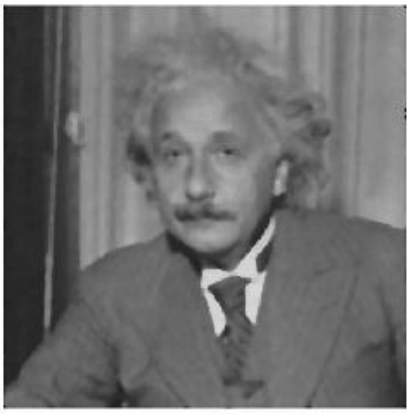




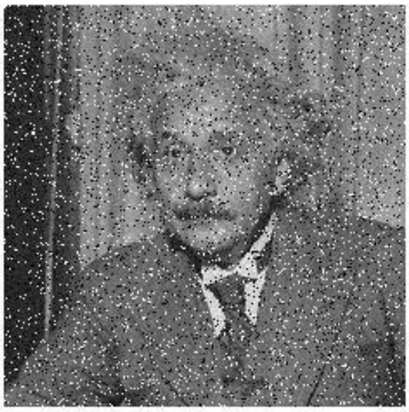


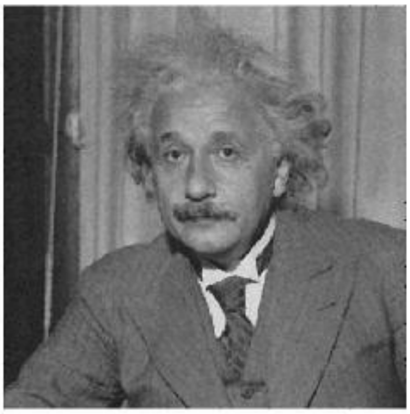


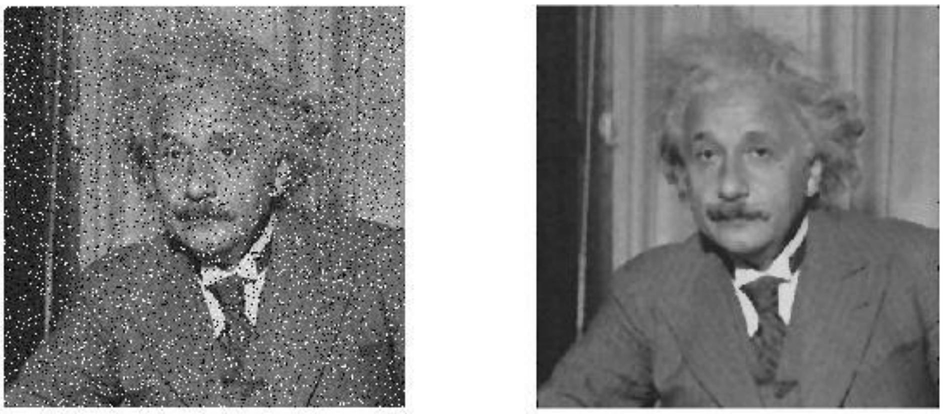


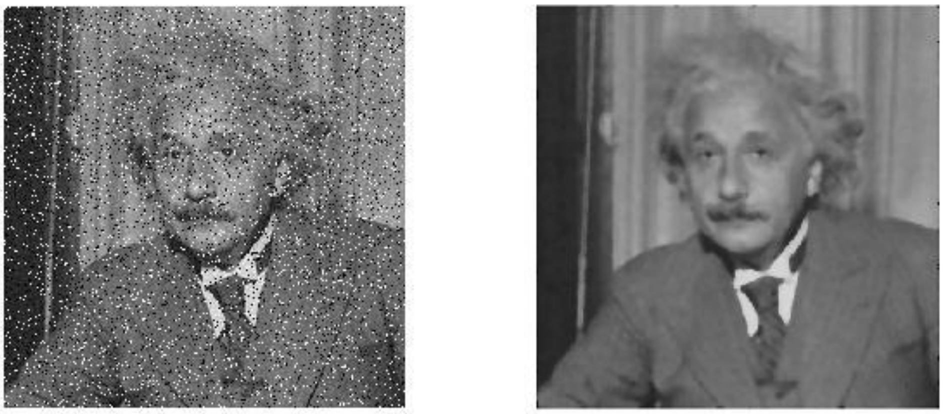


einstein3 with 3, 5 and 7 median filters:

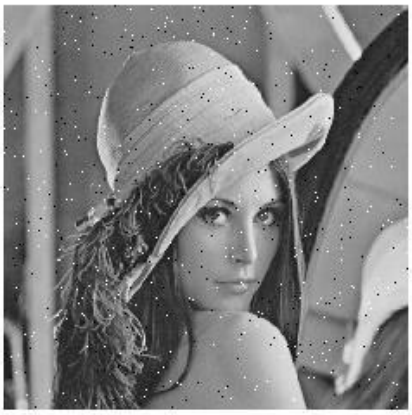


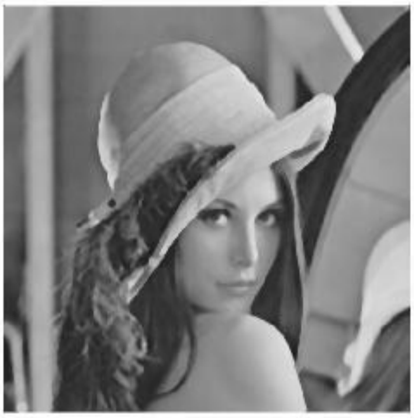




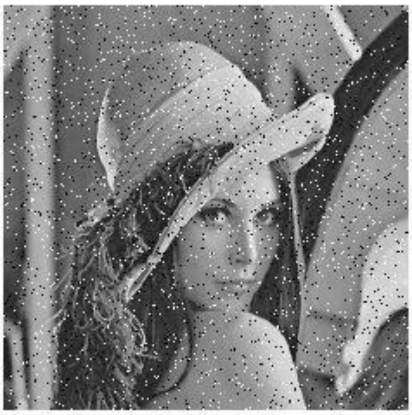


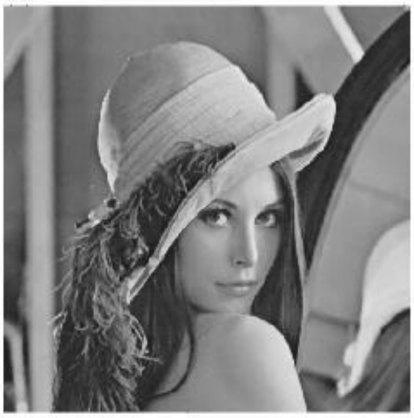
Lena1 with 3, 5 and 7 median filters:



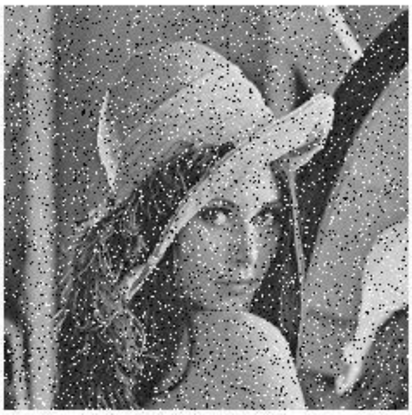


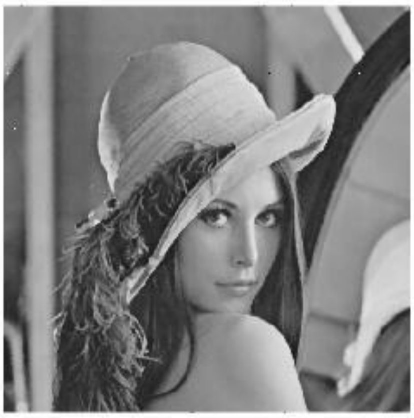
Lena2 with 3, 5 and 7 median filters:

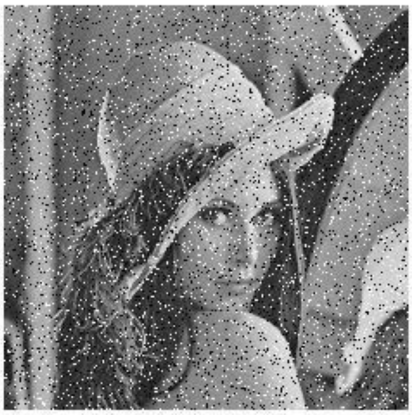


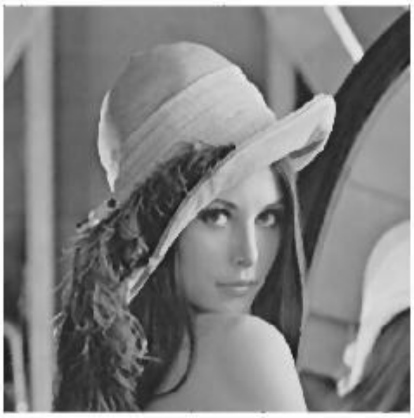


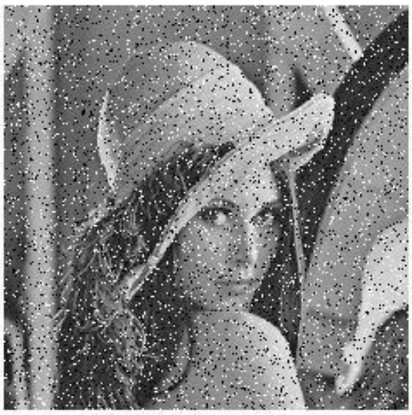
Lena3 with 3, 5 and 7 median filters:

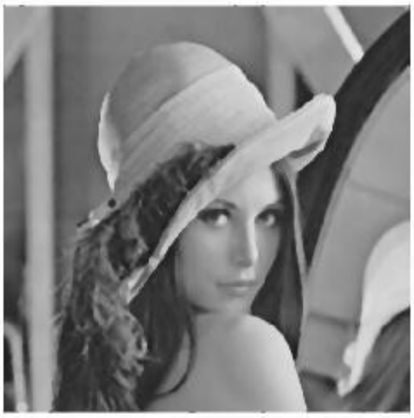






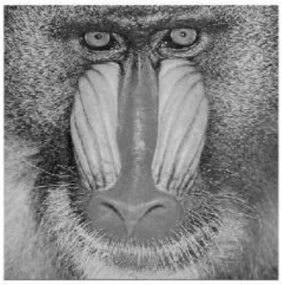


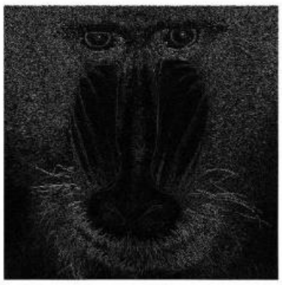


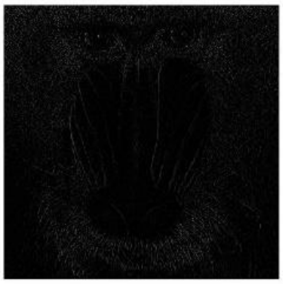


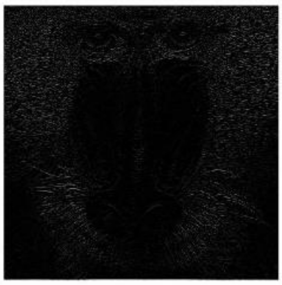
2) I implemented two different scripts for this problem. Both result the same. One of them uses for loops to apply the given filter, the other one uses conv2 function to do it. Results include x-filtered, y-filtered and gradient images.

Baboon:

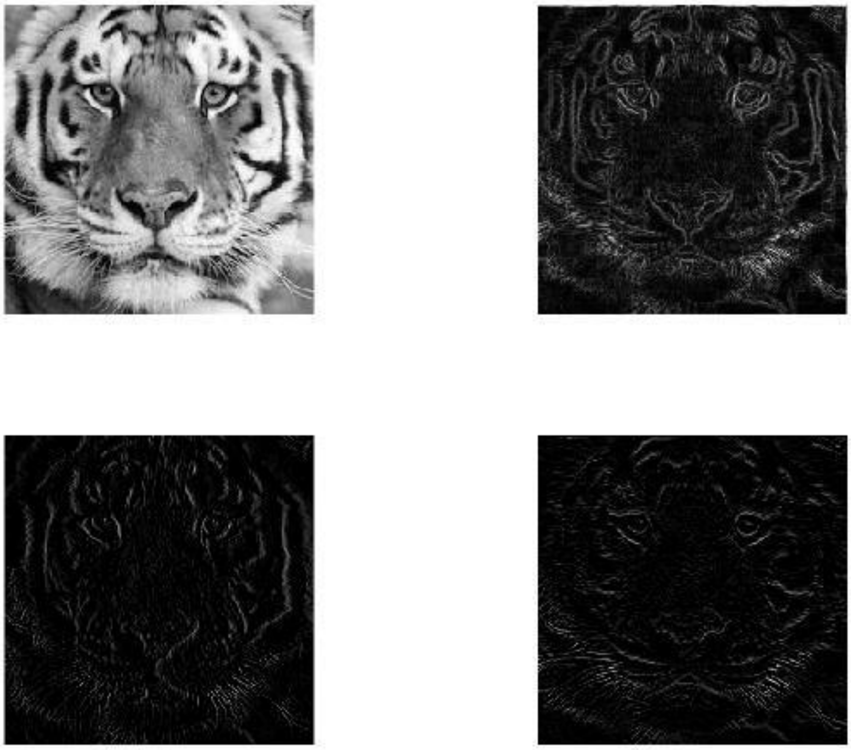








Lion:



Zebra: