CPSC 425 Assignment 1

Name: Osman Hajiyev
Student #: 54667143

Part 1 Question 1:

boxfilter(5):
```
[[ 0.04  0.04  0.04  0.04  0.04]
 [ 0.04  0.04  0.04  0.04  0.04]
 [ 0.04  0.04  0.04  0.04  0.04]
 [ 0.04  0.04  0.04  0.04  0.04]
 [ 0.04  0.04  0.04  0.04  0.04]]
```

boxfilter(4):

```
AssertionErrorTraceback (most recent call last)
/Users/osman/School/CPSC425/a1/a1.py in <module>()
     19 print(boxfilter(5))
     20 print("boxfilter(4):")
---> 21 print(boxfilter(4))
     22 print("boxfilter(3):")
     23 print(boxfilter(3))
/Users/osman/School/CPSC425/a1/a1.py in boxfilter(size)
      6 #returns a 2D box filter of size - size X size
      7 def boxfilter(size):
----> 8     assert size % 2 != 0, "Dimension must be odd!"
      9     assert size > 0, "Dimension must be positive!"
     10     return np.full((size, size), 1.0/(size*size))
AssertionError: Dimension must be odd!
```

boxfilter(3):
```
[[ 0.11111111  0.11111111  0.11111111]
 [ 0.11111111  0.11111111  0.11111111]
 [ 0.11111111  0.11111111  0.11111111]]
```

Question 2:

gauss1d(0.3):
```
[0.0038362587991689332, 0.99232748240166202, 0.0038362587991689332]
```
gauss1d(0.5):
```
[0.10650697891920073, 0.7869860421615984, 0.10650697891920073]
```

gauss1d(1):

```
[0.0033357736406091497, 0.067000804625851229, 0.18212706970658615,
0.49507270405390696, 0.18212706970658615, 0.067000804625851229,
0.0033357736406091497]
```

gauss1d(2):

```
[0.0021998111785770007, 0.005979706752766936, 0.044184388651349327,
0.044184388651349327, 0.12010562077253494, 0.12010562077253494,
0.32648092644177495, 0.12010562077253494, 0.12010562077253494,
0.044184388651349327, 0.044184388651349327, 0.005979706752766936,
0.0021998111785770007]
```

Question 3:

gauss2d(0.5):

```
[[ 0.01134374  0.08381951  0.01134374]
 [ 0.08381951  0.61934703  0.08381951]
 [ 0.01134374  0.08381951  0.01134374]]
```

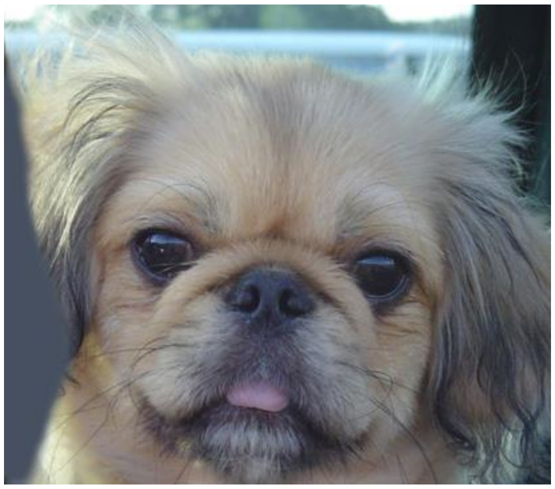
gauss2d(1):

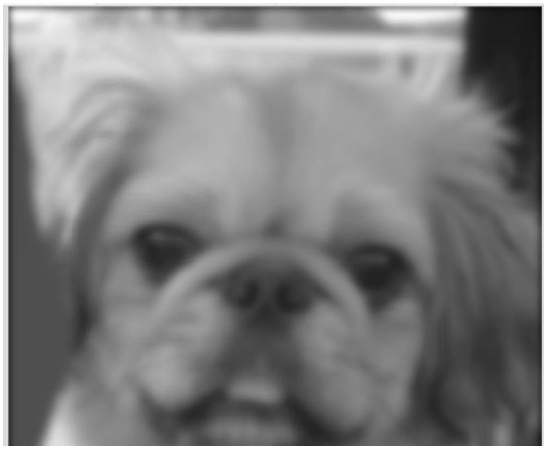
```
[[  1.11273858e-05   2.23499518e-04   6.07534678e-04   1.65145048e-03
    6.07534678e-04   2.23499518e-04   1.11273858e-05]
 [  2.23499518e-04   4.48910782e-03   1.22026602e-02   3.31702695e-02
    1.22026602e-02   4.48910782e-03   2.23499518e-04]
 [  6.07534678e-04   1.22026602e-02   3.31702695e-02   9.01661409e-02
    3.31702695e-02   1.22026602e-02   6.07534678e-04]
 [  1.65145048e-03   3.31702695e-02   9.01661409e-02   2.45096982e-01
    9.01661409e-02   3.31702695e-02   1.65145048e-03]
 [  6.07534678e-04   1.22026602e-02   3.31702695e-02   9.01661409e-02
    3.31702695e-02   1.22026602e-02   6.07534678e-04]
 [  2.23499518e-04   4.48910782e-03   1.22026602e-02   3.31702695e-02
    1.22026602e-02   4.48910782e-03   2.23499518e-04]
 [  1.11273858e-05   2.23499518e-04   6.07534678e-04   1.65145048e-03
    6.07534678e-04   2.23499518e-04   1.11273858e-05]]
```

Question 4:

4a. They are same in 2d gaussian case because 2d gaussian is symmetric whereas anything that is not symmetric will most likely have different answers, and these functions do completely different different things.

4b and c. Images attached below

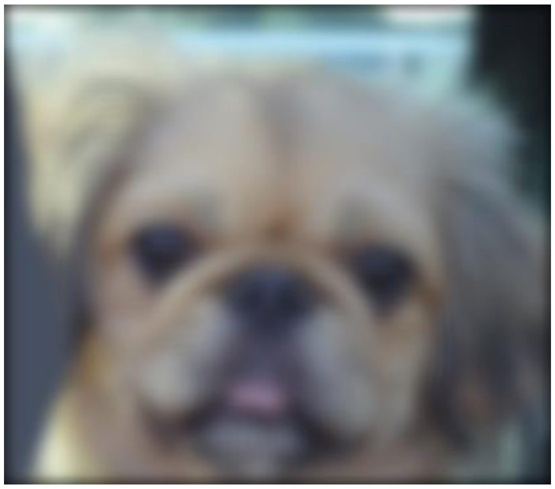

Original image

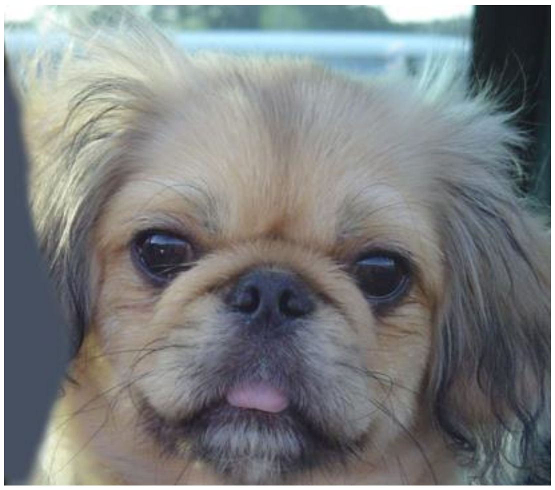

Result image

Question 5:
5. We can divide the formula of 2D gaussian (x and y) filter into separate x part of the filter and y part of the filter. As a result we will have simpler and faster (2m*n^2) operations because we only need to filter horizontally, then vertically and then add them up, where as with 2D gaussian filter we will have to over every element and it will be slower. (m^2*n^2). where n is the number of pixels and m is the number of multiplications needed to apply the filter.

Part2:

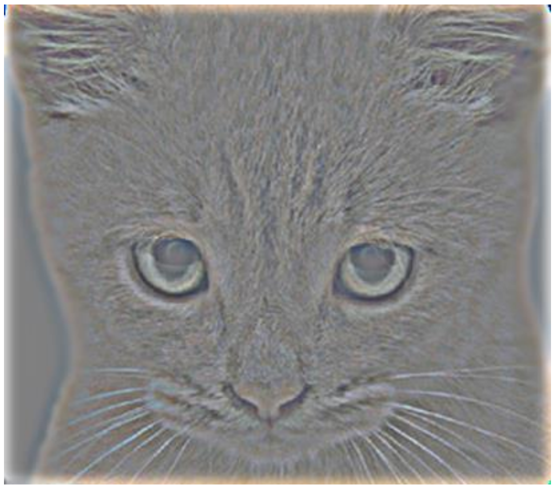
Question 1:
Below attached:


result Image


Original Image

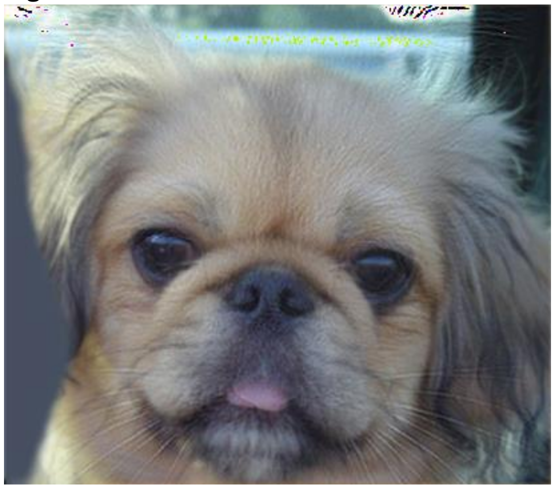
Question 2:

Original Image

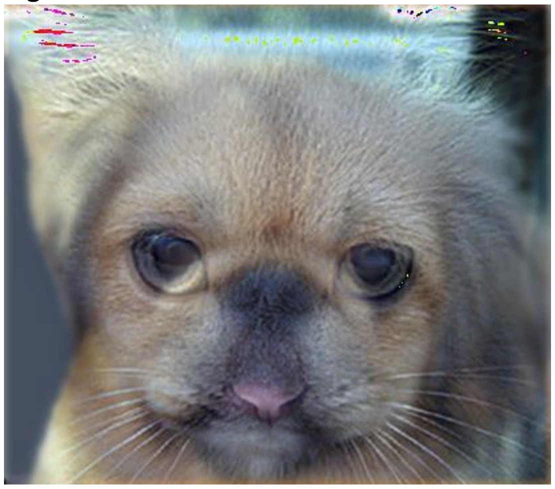
 Result Image

Question 3: 3 Images attached at the bottom of PDF for values sigma=1, sigma=3 and sigma=5

Sigma 1:



Sigma 3:



Sigma 5: