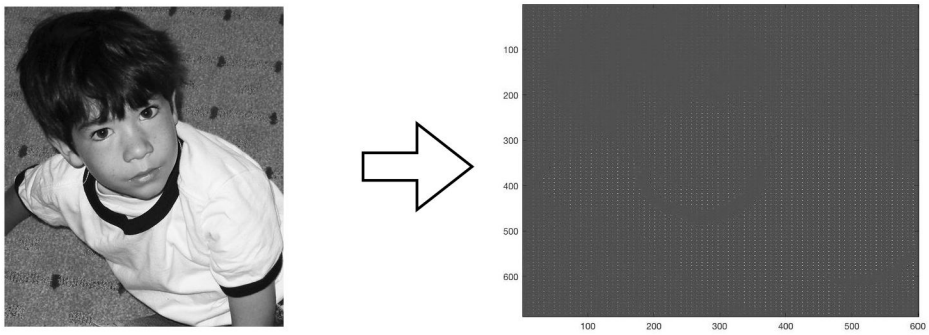
4. Approximating baseline jpeg

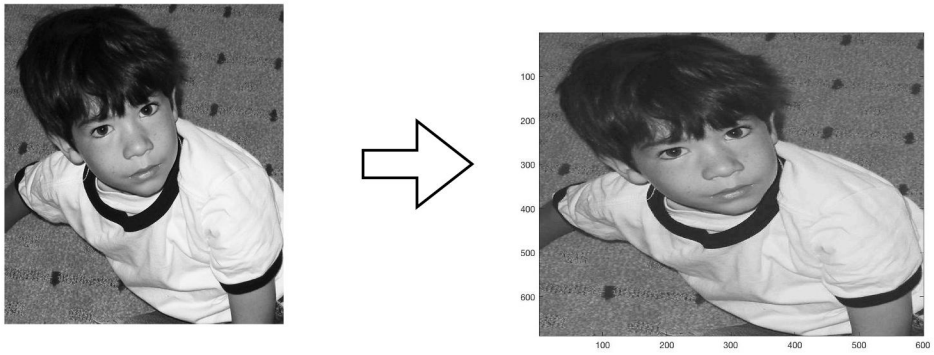
a) Write a function g=blockDCT(f, blockSize, q) to block DCT encode image f using blocks of size blockSize (e.g. 8 for 8x8 block size) and quantize the q result using matrix q. Assume q is the same size as specified by the block size.




```
%perform dct on block of 8 by 8
dct = @(block_struct) T * block_struct.data * T';

%performing quantization
c = @(block_struct)(block_struct.data) ./ q_mtx;
```

b) Write a function h=blockIDCT(g, blockSize,q) to reconstruct the original image from the DCT block coded matrix g (blockSize and q similar to part a).




```
%performing inverse DCT on block of 8 by 8
invdct = @(block_struct) round(T' * block_struct.data * T);
I2 = blockproc(B3,[8 8],invdct);
```

c) Write a fuction h=blockEntropy(g, blockSize) to calculate the average block entropy of g. It should calculate the entropy of each block by zig-zag scanning the block and calculating the entropy of the resulting 1d array.

```
%comparing the size of original and
compressed image
compression_ratio= (688*600*8)/length(I2)
```

E =

0.3995

d) Write a function c=compressionRatio(f, q) to calculate possible compression ratio of image f using block DCT coding with quantization matrix q (use functions you implemented in part a and c). Assume the uncompressed image takes 8 bits per pixel.

compression_ratio =

20.0254

e) Calculate the compression ratio and PSNR of the image 'boy.tif' using 8x8 blocks with the following quantization table. Decode the transformed image using the function you implemented in b and display the results.

PSNR =

34.2118

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$