# CS4551 Multimedia Software Systems (Spring 2018)

## Homework2 (5%) – Aliasing and Dictionary Coding

- Due: Electronic submission via CSNS by Monday, 03/12/2018.
- What to turn in:
  - Submit ***source code only.*** You must submit all files necessary for compile and run.
  - Do NOT submit data files.
  - You **MUST** provide a ***readme.txt*** file containing all information to help with the grading process.
- If your program produces any compile errors, you will receive 0 automatically no matter how close your program is to the solution.
- **You are not allowed to use any Java built-in image class methods, library, or tools to complete this homework.**
- You will receive penalty if you do not follow any of the given instructions.

## 0. What your program should do

Name your main application ***CS4551_[YourLastName].java*** (eg. CS4551_Doe.java) and execute it.

**<eg> On Command Prompt**

```
java CS4551_Doe
```

Display the following main menu to the user.

```
Main Menu-----------------------------------
1. Aliasing
2. Dictionary Coding
3. Quit

Please enter the task number [1-3]:
```

After performing a selected task, display the menu again.

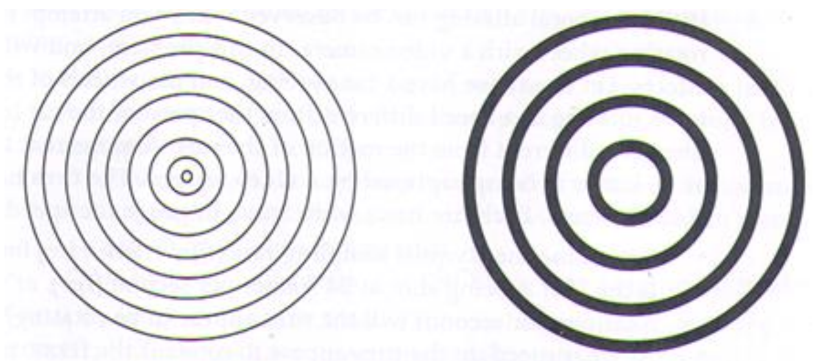
## 1. Task1 –Aliasing (50 pts):

Aliasing effects are observed for all kinds of media types. Here, we want to study aliasing effects in images.

- Step1: Create and display a 512x512 image containing a radial pattern of circles.

  A pattern of circles is centered at the center of the image and has two parameters describing it – M and N where M is the thickness of each circle in pixels and N is the difference between their successive radii in pixels.

Here are two sample images of M=1, N=5 and M=4, N=10 respectively.



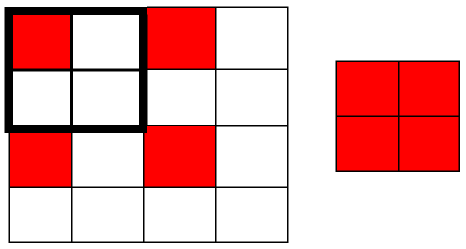
Receive values for M and N from the user to create the input image (512x512 image with white background and a radial pattern of black circles). Display the generated input image.

- Step2: Subsample the image by K (i.e. reduce the image size by K).

  Assume that K will be multiple of a power of 2 and $2 \leq K \leq 16$. When you subsample by K, your output image size will be 512/K by 512/K. For example, for K=4, your image size should be 128x128.

  According to K, resize the input image using the following three methods. ***Display three resized images.***

  1) Resize the input without any filer – Divide the original image into KxK blocks and pick the upper left corner of each block. **The total number of KxK blocks is (512/K)x(512/K) that is equivalent to the size of the shrink image. Use the upper left corner pixels** for the resize image pixels. For example, K=2, a 4x4 image is reduced to 2x2 as shown below.



  2) Resize the input with the 3x3 FILTER1 - Divide the input image into KxK blocks and locate the upper left corner of each block. Apply the FILTER1 to the pixel. Use the filtered pixels for the resize image pixel.
  3) Resize the input with the 3x3 FILTER2 - Divide the input image into KxK blocks and locate the upper left corner of each block. Apply the FILTER2 to the pixel. Use the filtered pixels for the resize image pixel.

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

FILTER1

| | | |
|---|---|---|
| 1/16 | 2/16 | 1/16 |
| 2/16 | 4/16 | 2/16 |
| 1/16 | 2/16 | 1/16 |

FILTER2

Test your program using the following values. Report your finding (which method is working the best and why you think) in the readme.txt file.

```
M=1, N=20, K=2
M=1, N=20, K=4
M=3, N=20, K=2
M=3, N=20, K=4
M=5, N=40, K=2
M=5, N=40, K=4
```

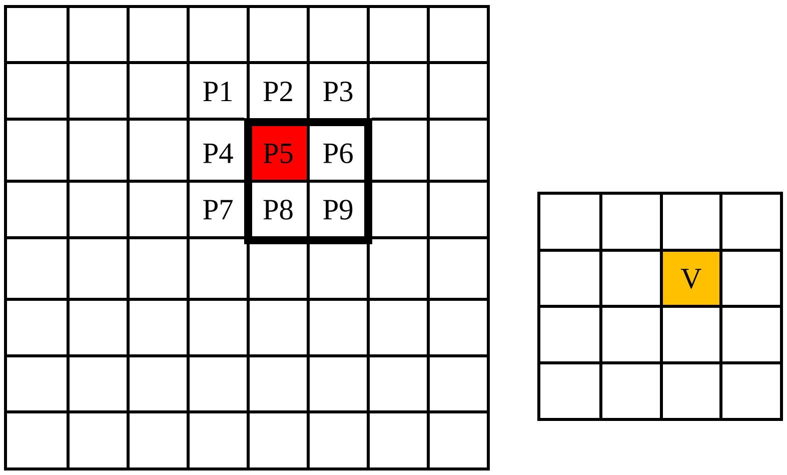
*** Applying a 3x3 filter to a pixel P(x,y)**

Let's assume that we have a 3x3 filter with filter values F1~F8 and C as shown below.

| | | |
|---|---|---|
| F1 | F2 | F3 |
| F4 | C | F5 |
| F6 | F7 | F8 |

The filtered value of P(x,y) is computed by

```
V = filter(P(x,y)) =  P(x-1, y-1)*F1 + P(x,y-1)*F2 + P(x+1, y-1)*F3 +
                      P(x-1, y)  *F4 + P(x,y)  * C + P(x+1, y)  *F5 +
                      P(x-1, y+1)*F6 + P(x,y+1)*F7 + P(x+1, y+1)*F8
```



**`P1 = p(x-1, y-1), P2 = P(x,y-1), P3 = P(x+1, y-1), … , P5 = P(x,y), …`**

See Circle Sample Results.

## 2. Task 2 – Dictionary Coding (50pts)

Write an encoder and decoder implementing the dictionary coding algorithm.

1) Encoder

As input, your program should take a user-defined character string from a file. Accept the file name from the user. Assume that all symbols in the file are 8-bit characters.

Calculate the actual dictionary size and determine a precise number of bits needed for the dictionary indices. Assume that the maximum dictionary size is 256 (8 bits). See the LZW algorithm lecture note for an example and check how the data size after compression is computed.

As output, display to the console 1) the generated dictionary, 2) a list of indices to encode the data, and 3) compression ratio. See below for the compression ratio calculation.

*Compression ratio = (Data Size before Compression) / (Data Size after Compression)*

*Data Size before Compression = (number of bits per symbol) x (number of symbols in the input).*

*Data Size after Compression = (number of bits to represent the unique entries in the dictionary) x (number of indices generated to encode the input sequence)*

To calculate the data size before compression, assume 8 bits per symbol.

2) Decoder

Decoding should be done. Perform decoding immediately after encoding and display the decoded message. Your source code will be checked for this part.

See LZW test data and a sample result.