Objective:

- Surfing an application of matrices and array representation scheme.
- Defining mappings for different Arrays representations.

Task – 1: ***ND-Array***

Define a function which prints on console the Row-Major/Column-Major mapping for a given number of dimensions.

```
enum ArrayMapping {ROW_MAJOR, COLUMN_MAJOR};
void printND(int N, ArrayMapping = ROW_MAJOR);
```

Examples:

```
Call to printND(3)                  prints      I1*D2*D3 + I2D3 + I3
Call to printND(1)                  prints      I1
Call to printND(2, ROW_MAJOR)       prints      I1D2 + I2
Call to printND(4, COLUMN_MAJOR)    prints      I4*D3*D2*D1 + I3*D2*D1 + I2*D1 + I1
```

Where $(i_1, i_2, i_3, \ldots, i_N)$ represents the index set and $(d_1, d_2, d_3, \ldots, d_N)$ represents the dimension set.

Note: Your function output must match with the console output shown above.

Task – 2: ***Array Mapping***

1. Develop a 1D array mapping for a language in which first index start from 3 but underlaying array in RAM has starting index 0.

2. Develop a 1D array mapping for a language in which first index start from 'k' but underlaying array in RAM has starting index 's'.

3. Develop a 1D array mapping for a language in which first index start from 'k' but underlaying array has starting index 's' and its each index has a gap of 'g'.

```
For Example: for array, int a[6];// Assume k=5, g=2, and s=7
    Map the array (as perceived by programmer)indices are: 5, 6, 7,  8,  9,  10  TO
                                        Underlaying Array: 7, 9, 11, 13, 15, 17
```

4. Develop a row-major order formula for 2D array where underlaying 1D array index start from 0 but for a programmer the 1st dimension starts from 's' and 2nd dimension start from 'k'.

```
For Example: int a[3][4]; // Assume s=-1, k=5
The first element in matrix will be a[-1][5]
For Programmer: array(row, col)     1st row will be:  (-1,5), (-1,6), (-1,7), (-1, 8)
                                    2nd row will be:  (0,5) , (0,6) , (0,7),  (0, 8)
                                    3rd row will be:  (1,5) , (1,6) , (1,7),  (1, 8)
Underlaying Array: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

So, (-1,5) maps to 0, (1,6) maps to 9
```

Task – 3: ***Array Mapping***

An image is nothing but a matrix. What is the meaning of this! Let's learn.
To understand this, we first need to know the basics of color. We know that every pixel/color is a combination of three basic colors I.e. Red, Green, and Blue. By changing the intensity of three basic colors from 0 ~ 255 we produce different colors. So a pixel in computer, which represents a color basically, contains three integral values one for red, second for green and last for blue.
So an image is a collection of pixels, which are arranged in matrix form.
For example figure below shows the color data of an image whose dimension is just 4 by 4. It means that there are only 16 pixels in the image.
So, the element at [0][0][0], [1][0][0], and [2][0][0] forms one pixel and element at [0][0][1], [1][0][1], and [2][0][1] will form another pixel and so on ... In short if we overlap these three matrices; we get finally 16 pixels in matrix form representing an image.

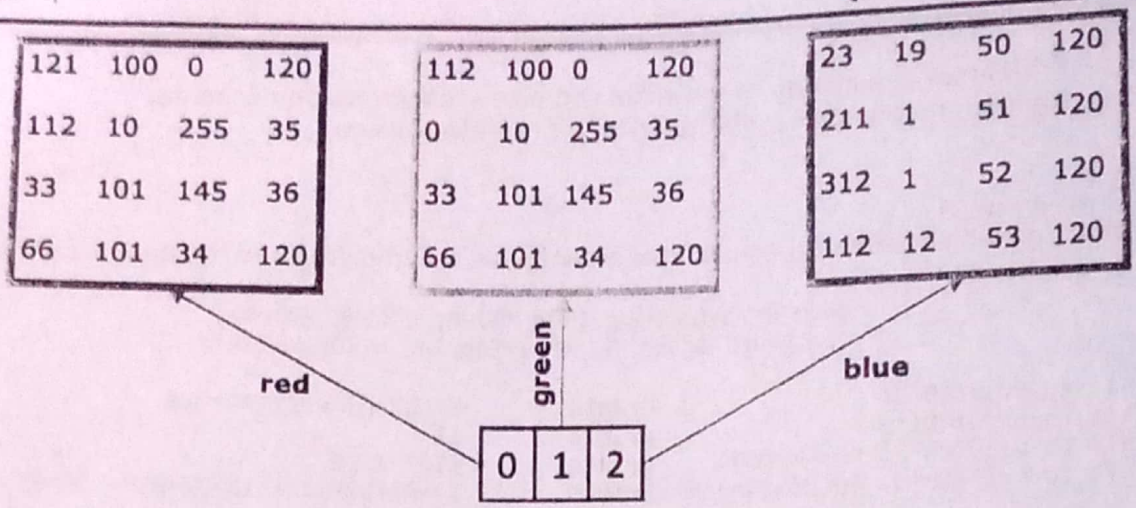

Figure-A

The way an image data layout (more specifically the pixel information) is represented is called Format of an image. You may have seen different image formats i.e. jpeg, gif, png etc.
The example shown above is also an image format called 'PPM' which is an uncompressed format (Hit the Google if you want to learn more about it)

So after getting some basics of an image answer the following:

When a PPM image is stored in a file (on secondary storage), it is not stored as it is shown in above diagram rather it is stored as follows:
the first line of the file shows the image dimension (rows*cols) and below this we have pixel information.

```
4 4
121 112 23 100 100 19 0 0 50 120 120 120 112 0 211 .....
```

Figure-B

So, the first red, green and blue maps to [0][0][0], [1][0][0], [2][0][0] respectively and second red, green and blue values maps to [0][0][1], [1][0][1], [2][0][1] and so on.

Let's assume that the image data is a 1-D array as it is shown in the file.
So answer the following questions:

1) Give a $O(1)$ mapping scheme which maps the pixel information of figure-A to figure-B. I.e. a given (i_1, i_2, i_3) is represented by which byte number ***N*** in the 1-D array. You can consider the first red value as byte number 1 and the green value as byte number 2 and so on.

2) Also give a $O(1)$ mapping scheme which is a vice versa of the first one. I.e., the mapping scheme, which maps the given byte number ***N*** in figure-B to (i_1, i_2, i_3) in figure-A.