Objective:

- This quiz will check your status of learning so far in the course by asking questions from easy to difficult based on knowledge learned so far.

Challenge-A: (4,4,2)

Give worst **time** and **space** bound of the following code function?

Part-A

```
void silly(int N, int x, int y)
{
    int * * p = new int * [ N ];
    for (int i=0; i<N; i++)
        p[i] = nullptr;
    if (x < y)
    {
        for (int i = 1; i <= N; i++)
        {
            int k=0;
            for (int j = 1; j <= N * i; j++)
            {
                p[k] = new int[N];
                k = (k+1)%N;
                if (k==0)
                {
                    int s=0;
                    while(s<N && p[s]!=nullptr)
                    {
                        delete [] p[s];
                        p[s] = nullptr;
                        s++;
                    }
                }
            }
        }
    }
    else
        cout<<"bye";
}
```

Part-B

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 a pixel. We know that every pixel 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.

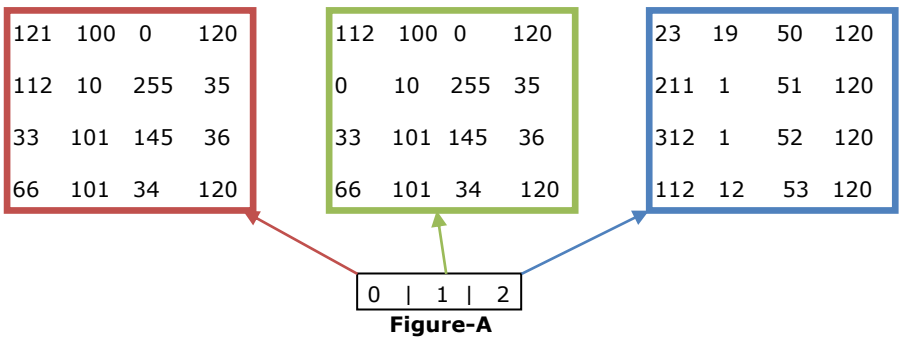

Figure-A

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 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 (an image format) 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. In the PPM file, each pixel is defined by a triplet of values representing how much red, green, and blue (RGB) are present. So, the pixel, which has a value of 0 0 0, is black, and the pixel, which has a value of 255 255 255, is white. By varying the levels of the RGB values you can come up with any color in between.

```
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], and [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.
Based on the above discussion, 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][j][k] is represented by which byte number in the 1-D array for a given image of dimension *R* by *C*. 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 'K' in figure-B to [i][j][k] in figure-A.

Part-C

For any given infix expression; while converting it from infix to postfix using Expression Evaluation Algorithm; the operator stack status will ever be as follows? : [True/False]
For each case, Stack is growing from left to right and if you feel stack status is valid then answer in true otherwise false.

CASE-A:	/(/	**ANSWER: TRUE ___ FALSE ___**
CASE-B:	*/	**ANSWER: TRUE ___ FALSE ___**
CASE-C:	/(+	**ANSWER: TRUE ___ FALSE ___**
CASE-D:	/+ (+	**ANSWER: TRUE ___ FALSE ___**

Challenge-B: **(7)**

A polynomial of degree n is a function of the form

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

where the $a's$ are real numbers (sometimes called the coefficients of the polynomial). Although this general formula might look quite complicated, particular examples are much simpler.
For example, $f(x) = 4x^5 - 3x^2 + 2$ is a polynomial of degree 5, as 5 is the highest power of x in the formula.

You task is to write an efficient function which finds the value $f(x)$.
For example, for the equation $f(x) = 4x^5 - 3x^2 + 2$, the $f(2) = 128 - 12 + 2 = 118$, Your function will return 118.

Function Prototype

```
double evaluatePolynomial(double coefficients[], int exponents[], int N, int x);
```

Polynomial equation will be stored using two arrays of size N. where N is the number of terms in the equation.
For Example: to store the equation $4x^5 - 3x^2 + 2$, The following arrays will be used:

```
double coef [] = {4, -3, 2};
int expo[] = {5, 2, 0};
cout << evaluatePolynomial (coef, expo, 3, 2); // prints 118 on console.
```

Assumptions:

- Exponents array will be sorted in descending order.
- The coefficient of `exponents[i]` will be stored at `coefficients[i]`.
- All the terms will be unique. It means, two terms cannot have same exponent.

You are also required to provide time and space equation of your code by providing the step count of each statement.

Challenge-C: **(5)**

This task is our first step towards indenting our C/C++ code automatically.
In this initial task, you have to handle/indent the C++ blocks i.e. curly braces only. So, you may ignore rest of the code.
You have to implement the function '*indentCode*', which receives a code file name and produces another code file which contains indented code. Keep in mind the time/space cost of your solution.

```
void indentCode ( string & fileName );
```

Note:

- You are not allowed to change anything in the received file.
- If the received file name is "abc.cpp" then the output file name must be "indented_abc.cpp".
- You will use white space character *tab* to indent the curly braces.

Assumption:

- There will be no syntax error in the input file as per C++ language rules.

Look at the example below and input/output files (Sample Input/Output) sent with this task to understand further the purpose of this task.

Example:

Input File Name: "Test 1.cpp"

```
#include <iostream>
using namespace std;

int f(int a) {cout<<"bye";return a;}
int main()
{  {{}}









    {
                        cout<<"dgff";}
    return 0;}
```

Output File Name: "indented_Test 1"

```
#include <iostream>
using namespace std;

int f(int a)
{cout<<"bye";return a;
}
int main()
{
        {
                {
                }
        }
```

```
        {
                cout<<"dgff";
        }
    return 0;
}
```

Challenge-D: **(5)**

You are required to right a function '`void displayPrimes(int N)`', which display prime numbers up to given integer 'N'.
The logic that you are required to use to implement '`displayPrimes`' function is based on an algorithm named as "Sieve of Eratosthenes", which is used to find prime numbers up to 'N'. The algorithm is described below.
***Algorithm*:** for finding prime numbers up to N.

- Create a queue and fill it with the consecutive integers 2 through N inclusive.
- Create an empty queue to store primes.
- limit = sqrt(N);
- do
  {
  - Obtain the next prime p by removing the first value in the queue of numbers.
  - Put p into the queue of primes.
  - Go through the queue of numbers, eliminating numbers divisible by p.

  }
  while (p < limit);
- all remaining values in numbers queue are prime, so transfer them to primes queue

Note: You don't need to provide Queue ADT implementation: You can use the same Queue ADT that we discussed in class/lab.
Note:

- You are allowed to library sqrt function if needed.

You have to write two versions of the above function:

- using user/programmer define structures only if needed.
- using STL structures only if needed.

General Guidelines and Submission Instructions:

- How to Submit:
  - You will submit **pdf** file of your answer on the piazza privately.
    - You must tag your post with online_quiz_2.
  - Your File name will be your RollNo-Quiz-2.
  - You have to mention your Name/RollNo/Quiz Title within the answer file first.
- A reminder:
  - No help from internet/books/humans/machines etc. allowed in order to achieve your answers. You have to do it at your own.
- Relaxation was given in your previous online quiz but this time, your quiz may not be graded in case of:
  - Late submission
  - Multiple submissions or post editing.
  - Failing to meet any of the submission instructions.