Problem 1: Count Operations (20 points)

a) **(10 points)** What is the number of operations executed when the following Java code runs? The rules for counting operations in Pseudocode apply for Java code, `StdOut.print()` and `StdIn.readInt()` are equivalent to DISPLAY and READ respectively.

```
public class AddThree
{
   public static void main(String[] args)
   {
      StdOut.print("Type the first integer: ");
      int x = StdIn.readInt();
      StdOut.print("Type the second integer: ");
      int y = StdIn.readInt();
       StdOut.print("Type the third integer: ");
      int z = StdIn.readInt();

      int sum = x + y + z;
      StdOut.print("Their sum is " + sum);
   }
}
```

a) **(10 points)** What is the number of operations executed when the following Java code runs? Your answer is expected to be in terms of `n`. The rules for counting operations in Pseudocode apply for Java code, `StdOut.println()` and `Integer.parseInt(args[0])` are equivalent to DISPLAY and READ respectively.

```
public class RandomSequence
{
   public static void main ( String[] args )
   {
      int n = Integer.parseInt(args[0]);
      for (int i = 0; i < n; i++)
      {
         StdOut.println(Math.random());
      }
   }
}
```

Problem 2: IF Conditional (20 points)

a) **(8 points)** Write a Java program called Div that takes 2 (two) double command-line arguments as inputs, dividend and divisor (in that order) and performs a division operation. Your program either prints the quotient or an error if the divisor is zero. The divisor is the number you divide the dividend by.

```
public class Div
{
    public static void main ( String[] args )
    {
       // WRITE YOUR CODE HERE
    }
}
```

b) **(8 points)** Write a code fragment that examines the integer variable **x**, printing GT (greater than) if the integer is greater than 100, LT (less than) if the integer is less than 100 and EQ (equal) if the number is 100.

```
public class IntCheck
{
    public static void main ( String[] args )
    {
        int x = Integer.parseInt(args[0]);
       // WRITE YOUR CODE HERE
    }
}
```

c) **(4 points)** Which line of code below does NOT have an error:

```
a) if ( x > y ) then System.out.print("No Error");
b) if ( x > y ) { System.out.print("No Error") }
c) if ( y < x ) d = 4; else d = 6;
d) if ( y = x ) d = 4;
```

Problem 3: Loops (20 points)

a) (**10 points**) Describe what the following code calculates.

```
1  public class Example
2  {
3    public static main (String[] args)
4    {
5      int sum = 0;
6      int n = 1000;
7      for (int i = 1; i <= n; i++)
8      {
9        if (i % 10 != 6)
10         sum = sum + i;
11     }
12     System.out.println(sum);
13   }
14 }
```

b) (**10 points**) The following code is supposed to calculate $1/1^2 + 1/2^2 + 1/3^2 + \ldots + 1/9^2 + 1/10^2$. However, there is a bug. In order to fix it, we need add this code _____ in _____

```
1  public class Example2
2  {
3    public static main (String[] args)
4    {
5      double sum = 0;
6      int num = 1;
7      while (num < 10)
8      {
9        sum = sum + 1.0/(num*num);
10     }
11     System.out.println(sum);
12    }
13 }
```

Problem 4: 1D Arrays (20 points)

1. Consider the following Java code fragment. The numbers on the left are line numbers for reference only and are not part of the code.

```
1  int[] a = new int[n];
2  a[0] = 0; a[1] = 1;
3 int x = 0;
4  for (int i = 2; i < n; i++) {
5        x = 3*(i-1)*(i-1);
6        x += 3*i - 2;
7         a[i] += a[i-1] + x;
8    }
```

a) **(4 points)** Suppose n = 6. Complete the table below for i= 2, 3, 4, 5

i	x	a[i]
1	0	1

b) **(4 points)** explain briefly what the code does by observing the content of array a[i]

c) **(2 points)** The code can be simplified by replacing lines 5, 6, 7 with a single line of code. Write the single line of code.

Problem 5: 2D arrays (20 points)

a. Write a Java program that will read from `StdIn` an integer, `n`, and will set the size of a 2D array to have n rows and n columns. The values in this n x n 2D array must be read from `StdIn` and the array must be filled in row-major order.

For example, if the following input stream was read: 4 1 2 3 4 4 3 2 1 9 8 7 6 5 7 6 4
The 2D array created would be:

1	2	3	4
4	3	2	1
9	8	7	6
5	7	6	4

Include in the program above, code that would print the values in the 2D array beginning with the value in the last row, last column and ending with the value in the 0^{th} row, 0^{th} column, traversing the columns from bottom to top, printing n values in each row of the output. The values in the array do not change. You are simply printing them in a unique order.

For example, for the array given above, the output would be:

4	6	1	4
6	7	2	3
7	8	3	2
5	9	4	1