Exercise – Analysis of Algorithms

1. Count the number of operations and give the Big-O characterizations for the following Java code segments:

a.
```
public class StringExample
{
      public static void main(String[] args)
      {
            String s1 = "Computer Science";
            int x = 221;
            String s2 = s1 + " " + x;
            String s3 = s2.substring(10,17);
            String s4 = "is fun";
            String s5 = s2 + s4;

            System.out.println("s1: " + s1);
            System.out.println("s2: " + s2);
            System.out.println("s3: " + s3);
            System.out.println("s4: " + s4);
            System.out.println("s5: " + s5);

            //showing effect of precedence
            x = 3;
            int y = 5;
            String s6 = x + y + "total";
            String s7 = "total " + x + y;
            String s8 = " " + x + y + "total";
            System.out.println("s6: " + s6);
            System.out.println("s7: " + s7);
            System.out.println("s8: " + s8);
      }
}
```

b.
```
/* Illustrates how to call static methods
```

```
*  from within a method in the same class
*/

public class CallingMethodsInSameClass
{
      public static void main(String[] args)
      {
            printOne();
            printOne();
            printTwo();
      }

      public static void printOne()
      {
            System.out.println("Hello World");
      }

      public static void printTwo()
      {
            printOne();
            printOne();
      }
}
```

c. public class Factorial

```
{
      public static void main(String[] args)
      {
         final int NUM_FACTS = 100;
         for(int i = 0; i < NUM_FACTS; i++)
            System.out.println( i + "! is " + factorial(i));
      }

      public static int factorial(int n)
      {
            int result = 1;
            for(int i = 2; i <= n; i++)
                  result *= i;
            return result;
      }
}
```

d. //examples of array manipulations

```
public class ArrayExample
{
  public static void main(String[] args)
  {
     int[] list = {1, 2, 3, 4, 1, 2, 3};
     bubblesort(list);

      list = new int[]{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
      bubblesort(list);

      list = new int[]{11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
      bubblesort(list);


  }
}

/*
 *pre: list != null;
 *post: sort the elements of list into ascending order
 */
 public static void bubblesort(int[] list)
 {
      int temp;
      boolean changed = true;
      for(int i = 0; i < list.length && changed; i++)
      {
            changed = false;
            for(int j = 0; j < list.length - i - 1; j++)
            {
                  if(list[j] > list[j+1])
                  {
                        changed = true;
                        temp = list[j + 1];
                        list[j + 1] = list[j];
                        list[j] = temp;
                  }
            }
      }
 }
```

2. Given the following number of operations for some unspecified algorithms, give the Big-Oh notation of their growth rates:

   a. $n \log n + 10n + 1000$

   b. $500\ n^2 + 2^n$

   c. $500 \log n + n^3 + 300n + 7$

   d. $1{,}000{,}000 + 10 \log n + 5n$