Question – 1:

How many stars are displayed in the following code if n is 10? How many if n is 20?
Use the Big O notation to estimate the time complexity.

(a) for (int i = 0; i < n; i++) {
    System.out.print("*");
}

(b) for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print("*");
    }
}

(c) for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print("*");
        }
    }
}

(d) for (int k = 0; k < 10; k++) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print("*");
        }
    }
}

Question – 2:

Use the Big O notation to estimate the time complexity of the following methods:

(a) public static void mA(int n) {
    for (int i = 0; i < n; i++) {
        System.out.print(Math.random());
    }
}

(b) public static void mB(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            System.out.print(Math.random());
        }
    }
}

(c) public static void mC(int[] m) {
    for (int i = 0; i < m.length; i++) {
        System.out.print(m[i]);
    }
    for (int i = m.length - 1; i >= 0; )
    {
        System.out.print(m[i]);
        i--;
    }
}

(d) public static void mD(int[] m) {
    for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < m.length; j++) {
            System.out.print(m[i] * m[j]);
        }
    }
}
Question – 3:

Design an $O(n)$ time algorithm for computing the sum of numbers from $n1$ to $n2$ for $(n1 < n2)$. Can you design an $O(1)$ for performing the same task?

Question – 4:

Describe an algorithm for removing duplicates from an array. Analyze the complexity of the algorithm.

Question – 5:

Analyze the following sorting algorithm:

```java
for (int i = 0; i < list.length - 1; i++) {
    if (list[i] > list[i + 1]) {
        swap list[i] with list[i + 1];
        i = -1;
    }
}
```

Question – 6:

(Maximum consecutive increasingly ordered substring) Write a program that prompts the user to enter a string and displays the maximum consecutive increasingly ordered substring. Analyze the time complexity of your program. Here is a sample run:

```
Enter a string: abcabcgdgabxy
abcg
```

```
Enter a string: abcabcgdgbmnsxy
abmnxy
```