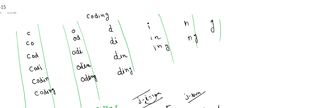
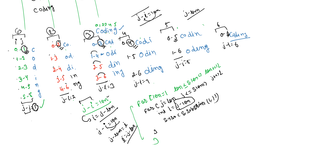
Lec-15





```
public static void print(String s) {
    for (int len = 1; len <= s.length(); len++) {
        for (int j = 0; j < s.length(); j++) {
            int i = j + len - 1;
            System.out.println(s.substring(j, i));
        }
    }
}
```

Karsh Bhaiya and Kanak Bhaiya are discussing a special type of number, which they call a **Coding Block Number (CB Number)**. A number qualifies as a CB Number based on the following criteria:

1. Odd
A CB number must be odd.

2. Divisible by 3
The following prime numbers are always CB numbers:
3, 5, 7, 11, 13, 17, 19, 23, 29, 31

3. Not a multiple of 5
Any other number is a CB number if it is not divisible by any of the numbers listed in point (2).

The Challenge

Karsh Bhaiya challenges Kanak Bhaiya with a problem:

Given a string of digits and since Bhaiya must determine the maximum number of CB numbers that can be extracted from it while following these constraints:

1. Non-overlapping Substrings
Each digit can be used only once.

2. CB Number cannot be a substring or a superstring of another chosen CB number.

Example: If 489 is both 489 and 891, its CB numbers, but we can choose only one of them.

3. Valid Substring Selection
The CB number must be a **contiguous** substring of the given string.

Example: In 891, we cannot select 81 as a CB number because 81 is not a contiguous substring of 891.

4. Maximization Goal
Since multiple solutions may exist, the goal is to find the **maximum number** of CB numbers that can be extracted from the given string.

```
public static void CB_Number(String s) {
    int c = 0;
    boolean[] visited = new boolean[s.length()];// false
    for (int len = 1; len <= s.length(); len++) {
        for (int j = 0; j < s.length(); j++) {
            int i = j + len - 1;
            int num = Integer.parseInt(s.substring(j, i));// j to i-1
            if (isCBNumber(num) && !isOverlapping(visited, j, i, len)) {
                for (int k = j; k < i; k++)
                    visited[k] = true;
                c++;
                System.out.print(num + " ");
            }
        }
    }
    System.out.print(c);
}

public static boolean isCBNumber(int[] visited, int i, int j) {// i to j-1
    // TODO Auto-generated method stub
    for (int k = i; k < j; k++) {
        if (visited[k] == true) {
            return false;
        }
    }
    return true;
}
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int a = 10;
    Integer b = 10;// auto-boxing
    System.out.println(a);
    System.out.println(b);
    int c = b;
    long l = 1000;
    int d = l;
    int[] ar = {1};// auto-boxing
    int e = ar[0];// un-boxing
}
```

```
public class ArrayList_Demo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ArrayList<Integer> ar = new ArrayList<>();
        ar.add(3);
        ar.add(2);
        ar.add(4);
        ar.add(1);
        System.out.
        ar.sort();
        System.out.
        ar.add(0, 5);
        System.out.
        ar.add(2, 6);
        System.out.
        ar.add(4, 7);
        System.out.
        ar.add(7, 8);
        System.out.
        ar.add(-2, 9);
        System.out.
        ar.add(-1, 10);
        System.out.
        ar.add(-3, 11);
        System.out.
        ar.add(-20, 12);
        System.out.
        ar.add(-20, 13);
        System.out.
        ar.add(-19, 14);
        System.out.
        ar.add(-18, 15);
        System.out.
        ar.add(-17, 16);
        System.out.
        ar.add(-16, 17);
        System.out.
        ar.add(-15, 18);
        System.out.
        ar.add(-14, 19);
        System.out.
        ar.add(-13, 20);
        System.out.
        ar.add(-12, 21);
        System.out.
        ar.add(-11, 22);
        System.out.
        ar.add(-10, 23);
        System.out.
        ar.add(-9, 24);
        System.out.
        ar.add(-8, 25);
        System.out.
        ar.add(-7, 26);
        System.out.
        ar.add(-6, 27);
        System.out.
        ar.add(-5, 28);
        System.out.
        ar.add(-4, 29);
        System.out.
        ar.add(-3, 30);
        System.out.
        ar.add(-2, 31);
        System.out.
        ar.add(-1, 32);
        System.out.
        ar.add(0, 33);
        System.out.
        ar.add(1, 34);
        System.out.
        ar.add(2, 35);
        System.out.
        ar.add(3, 36);
        System.out.
        ar.add(4, 37);
        System.out.
        ar.add(5, 38);
        System.out.
        ar.add(6, 39);
        System.out.
        ar.add(7, 40);
        System.out.
        ar.add(8, 41);
        System.out.
        ar.add(9, 42);
        System.out.
        ar.add(10, 43);
        System.out.
        ar.add(11, 44);
        System.out.
        ar.add(12,
```