Q 1: Which distance metric is usable for distances between keys?

A: Manhattan I think is more logical metrics when we are finding another keys

Q 2.1: Would you need a particular data structure to represent the keyboard layout?

A: Yes we need data structure to found the password

Q 2.2: Would this structure be needed permanent or once we calculate the distances between keys, could it be replaced by another structure?

A: I think we need to hold the data structure because If the coordinate of the keyboard change password might not be precise as before.

Q 3: Suppose we decided to map each key to a list of valid moves (ie. other keys with 2 to 3 distance). What kind of Java data structure be the best suited for this?
A: Not sure but Manhattan is still best I think.

Q 4: Write pseudocode (or Java code) for creating an 8 character password using the data structure you suggested.
A:

```
import java.util.*;

public class PasswordGenerator {


    private static final char[][] KEYBOARD = {
            {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'},
            {'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P'},
            {'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L'},
            {'Z', 'X', 'C', 'V', 'B', 'N', 'M'}
    };


    private static int[] findPosition(char c) {
        for (int i = 0; i < KEYBOARD.length; i++) {
            for (int j = 0; j < KEYBOARD[i].length; j++) {
                if (Character.toUpperCase(KEYBOARD[i][j]) == Character.toUpperCase(c))
{
                    return new int[]{i, j};
```

```
            }
        }
    }
    return null;
}


private static List<Character> getNearbyChars(char c) {
    int[] pos = findPosition(c);
    List<Character> nearby = new ArrayList<>();
    if (pos == null) return nearby;

    for (int i = 0; i < KEYBOARD.length; i++) {
        for (int j = 0; j < KEYBOARD[i].length; j++) {
            int distance = Math.abs(i - pos[0]) + Math.abs(j - pos[1]);
            if (distance >= 2 && distance <= 3) {
                nearby.add(KEYBOARD[i][j]);
            }
        }
    }
    return nearby;
}


public static String generatePassword(char startChar) {
    StringBuilder password = new StringBuilder();
    password.append(startChar);
    Random rand = new Random();

    char currentChar = startChar;
    while (password.length() < 8) {
        List<Character> options = getNearbyChars(currentChar);
        if (!options.isEmpty()) {
            currentChar = options.get(rand.nextInt(options.size()));
            password.append(currentChar);
        } else {
            break;
        }
    }
    return password.toString();
}

public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.print("İlk karakteri girin: ");
        char startChar = scanner.next().charAt(0);
        String password = generatePassword(startChar);
        System.out.println("Oluşturulan şifre: " + password);
    }
}
```

Q 5: Compute the list of valid moves for for the following keys: a, f, h, 8, 0, and p.

A: *Shown in the video*