III. Permutaciones sin Repeticiones

Ejercicio 1:

Genera todas las posibles permutaciones de un número divisible por N.

Referencia(s)

https://www.geeksforgeeks.org/generate-all-possible-permutations-of-a-number-divisible-by-n/?ref=rp

```
public static void getPermutationsDivisors(String s, int k) {
        LinkedList<Integer> L1 = new LinkedList<>();
        LinkedList<Integer> L2 = new LinkedList<>();
        for (int i = 0; i < s.length(); ++i) {
            L1.add(((int) s.charAt(i) - 48));
        }
        _permutSR(L1, L2, L1.size(), k);
    }

    private static boolean _isDivisible(LinkedList<Integer> L, int k) {
        String s = "";
        for (int i = 0; i < L.size(); ++i) {
            s += L.get(i).toString();
        }
        int d = Integer.parseInt(s);
        return (d % k == 0);
    }

    private static void _permutSR(LinkedList<Integer> L1, LinkedList<Integer> L2,
       int r, int div) {
        if (L2.size() == r && _isDivisible(L2, div)) {
            System.out.println(L2);
            return;
        }
        int k = 0;
        while (k < L1.size()) {
            if (!L2.contains(L1.get(k))) {
                L2.add(L1.get(k));
                _permutSR(L1, L2, r, div);
                L2.removeLast();
            }
            k = k + 1;
        }
    }
```

Ejercicio 2:

Encontrar las permutaciones posibles si se quiere buscar la contraseña perfecta con la palabra CLAVE.

```
public static void contrasenaSR(LinkedList<String> L1,LinkedList<String> L2, int
    r){
         if (L2.size()==r) {
             System.out.println(L2);
             c=c+1;
         }
         int k=0;
         while(k<L1.size()){
             if (!L2.contains(L1.get(k))) {
                 L2.add(L1.get(k));
                 contrasenaSR(L1, L2, r);
                 L2.removeLast();
             }
             k++;
         }
     }
```