IV. Permutaciones con Repeticiones

Ejercicio 1:

Mostrar todas las palabras pronunciables.

Referencia(s) https://www.tutorialspoint.com/print-all-permutations-with-repetition-of-characters-in-cplusplus

```
private static void _permutCR(LinkedList<Character> L1, LinkedList<Character> L2,
    int r) {
        if (L2.size() == r) {

            boolean ans = true;
            for (int i = 0; i < L2.size(); i++) {
                char c = L2.get(i);
                boolean isVowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' ||
                    c == 'U');
                if (!isVowel && i % 2 == 1) {
                    ans = false;
                }
            }
            if (ans) {
                System.out.println(L2);
            }
            return;
        }
        int k = 0;
        while (k < L1.size()) {
            L2.add(L1.get(k));
            _permutCR(L1, L2, r);
            L2.removeLast();
            k = k + 1;
        }
    }
```

Ejercicio 2:

Encontrar todas las permutaciones posibles segun la frecuencia de Letra que se encuentra en la palabra sin tomar en cuenta la frecuencia de la letra a buscar.

```
public static void PermSegunFrec(LinkedList<String> L1, LinkedList<String> L2, int
    cantRepe, String letra, int n, int r, int i) {
        if (L2.size() == r) {
            System.out.println(L2);
            c++;
            return;
        }
        int k = 0;
        while (k < n) {
            if (!L1.get(k).equals(letra) && cantRepe > 0) {
                L2.add(L1.get(k));
                PermSegunFrec(L1, L2, cantRepe - 1, letra, n, r, k);
                L2.removeLast();
            }
            k = k + 1;
        }
    }

    public static int cantRep(LinkedList<String> L1, String letra) {
        int c = 0;
        for (int i = 0; i < L1.size(); i++) {
            if (L1.get(i).equals(letra)) {
                c++;
            }
        }
        return c;
    }
}
```