Name: Alexandru-Gabriel PĂDUCEL

Critical & Logic rationale

Solution: The Coffee Cake is the first cake

Reasoning

What information is provided for us:

- We have three cakes;
- One of the cakes is a coffee cake;
- All three cakes all labeled;
- One of the labeled cake is labeled correctly and the other two are mislabeled;
- First cake label is “This is not he Coffee Cake”
- Second cake label is “This is not he Coffee Cake”
- Third cake label is “The Coffee Cake is in the Second Label”

From first and second cake label we know that one of the two is the Coffee Cake because only one label is correct as the other two are mislabeled.

The third label can’t be correct because if it was it would result that both first and the second cake are the Coffee Cake which is wrong as we only have one coffee cake.

The first label can’t be correct because, the second label would say that the second cake is the coffee cake and the third label would say that the second cake is not the coffee cake. The two label would contradict.

So the correct label is the second label, and that would mean the Coffee Cake is the first cake.

Algorithmic

The solution were implemented with JAVA.

1. Maximum of moves for the string s:

```
public int noMovesWithUnorderedLetters(String keyword, String word) {
    StringBuilder wordBuilder = new StringBuilder(word);
    StringBuilder keywordFromWord = new StringBuilder();

    int noMoves = 0;

    while (!wordBuilder.isEmpty()) {
        keywordFromWord.setLength(0);

        for (int i = 0; i < keyword.length(); i++) {
            String letter = String.
                    valueOf(keyword.charAt(i));
            int index = wordBuilder.indexOf(letter);
            if(index != -1)
            {
                keywordFromWord.append(wordBuilder.charAt(index));
                wordBuilder.deleteCharAt(index);
            }
            else break;
        }

        if (!keywordFromWord.toString().equals(keyword)) {
            break;
        } else noMoves++;
    }
    return noMoves;
}
```

Method call:

```
System.out.println("The number of aparitions of the word TIRAMISU in the
given word," +
        "TIRAXXMISU , is equal to "
        + new Solution().noMovesWithUnorderedLetters(
        "TIRAMISU",
        "TIRAXXMISU"));

System.out.println("The number of aparitions of the word TIRAMISU in the
given word," +
        "BATIRXXOLAMIXSU , is equal to "
        + new Solution().noMovesWithUnorderedLetters(
        "TIRAMISU",
        "BATIRXXOLAMIXSU"));

System.out.println("The number of aparitions of the word TIRAMISU in the
given word," +
        "TITIRHAUNRAOBGMIXZARBMISUSU , is equal to "
        + new Solution().noMovesWithUnorderedLetters(
        "TIRAMISU",
        "TITIRHAUNRAOBGMIXZARBMISUSU,"));

System.out.println("The number of aparitions of the word TIRAMISU in the
given word," +
        "QWERALE , is equal to "
        + new Solution().noMovesWithUnorderedLetters(
        "TIRAMISU",
        "QWERALE"));
```

Result:

```
The number of aparitions of the word TIRAMISU in the given word,TIRAXXMISU , is equal to 1
The number of aparitions of the word TIRAMISU in the given word,BATIRXXOLAMIXSU , is equal to 1
The number of aparitions of the word TIRAMISU in the given word,TITIRHAUNRAOBGMIXZARBMISUSU , is equal to 2
The number of aparitions of the word TIRAMISU in the given word,QWERALE , is equal to 0

Process finished with exit code 0
```

2. What should you change in order to find the number of occurrences for the reverse version of the word, USIMARIT?

Answer: Just calling the method from the first requirement with the parameter keyword equal to ‘USIMARIT’ would solve the problem as the method is written for every word.

3. How would you check if the letters occurrence is in the same order as in the given word?

```
public int noMovesWithOrderedLetters(String keyword, String word) {
    StringBuilder wordBuilder = new StringBuilder(word);
    StringBuilder keywordFromWord = new StringBuilder();

    int noMoves = 0;
    int indexOfKeyword;

    while (!wordBuilder.isEmpty()) {
        indexOfKeyword = 0;
        keywordFromWord.setLength(0);
        for (int i = 0; i < wordBuilder.length(); i++) {
            if (keyword.charAt(indexOfKeyword) == wordBuilder.charAt(i)) {
                keywordFromWord.append(wordBuilder.charAt(i));
                wordBuilder.deleteCharAt(i);
                i--;
                indexOfKeyword++;
            }
            if (indexOfKeyword == keyword.length()) {
                break;
            }
        }

        if (!keywordFromWord.toString().equals(keyword)) {
            break;
        } else noMoves++;
    }
    return noMoves;
}
```

Method Call:

```
System.out.println("The number of aparitions where the letters are in
order" +
        " where of the word TIRAMISU in the given word," +
        "TIRAXXMISU , is equal to "
        + new Solution().noMovesWithOrderedLetters(
        "TIRAMISU",
        "TIRAXXMISU"));

System.out.println("The number of aparitions where the letters are in
order" +
        " where of the word TIRAMISU in the given word," +
        "ITRAXXMISU , is equal to "
        + new Solution().noMovesWithOrderedLetters(
        "TIRAMISU",
        "ITRAXXMISU"));
```

Result:

```
The number of aparitions where the letters are in order where of the word TIRAMISU in the given word,TIRAXXMISU , is equal to 1
The number of aparitions where the letters are in order where of the word TIRAMISU in the given word,ITRAXXMISU , is equal to 0


Process finished with exit code 0
```

For the second call the result is 0 because the letters are not in order

4. What should you change in order to find the number of occurrences for any given keyword? Write down the implementation for this function. Implement a function that checks if within a 'basket' of strings (sugar, mascarpone, cocoa, coffee) you can find the basic ingredients to cook a tiramisu cake?

```
public boolean hasCorrectIngredients(String[] neededIngredients, String[]
ingredients) {
    HashMap<String, Integer> ingredientsExist = new HashMap<>();

    for (String neededIngredient : neededIngredients) {
        ingredientsExist.put(neededIngredient, 0);
    }

    for (String ingredient: neededIngredients) {
        for (String s : ingredients) {
            if (this.noMovesWithUnorderedLetters(ingredient, s) > 0) {
                ingredientsExist.put(ingredient, 1);
            }
        }
    }

    return ingredientsExist.values().stream().filter(num -> num ==
0).toArray().length == 0;

}
```

Method call:

```
String[] ingredients = new String[]{
        "CAOPCOXOLLA", "BAMASCARXXOLPOXNE",
        "QWCORAFFEE", "PLENLEGXP",
        "RPLSUORLEGXAPG", "QWERALE"
};
String[] neededIngredients = new String[]{
        "SUGAR", "MASCARPONE", "COFFEE", "COCOA"
};

if (new Solution().hasCorrectIngredients
        (neededIngredients, ingredients)) {
    System.out.println("We have all the ingredients for Tiramisu");
} else System.out.println("We don't have enough ingredients to make
Tiramisu");
```

Result:

```
We have all the ingredients for Tiramisu


Process finished with exit code 0
```

Technologies

What would you use to implement an API call and why?

If I have create a simple app I would use HttpURLConnection as is JAVA built-in so I don't need to install other dependencies to it. But if I have to create a more complex app I would use Apache HttpClient as supports HTTP2 and other modern HTTP features.

What techniques would you use to store data locally (in an Android app) and why?

If I have to save little information, that is not that important like user settings in the app I would use SharedPreferences, but I have to save more data like lists I would use SQLite.