Homework

4

Le, La, L', Les

Most of us realize that the rest of the world doesn't always use English when they name different things; even the names of the countries that appear on a map! Even if you though you knew the names of all of the states in the Union, you might be forgiven for stumbling at *Caroline du Sud*.

Both French and Russian are interesting because individual countries are **feminine** or **masculine**, depending on a complex set of rules. Russian is beyond me, but French country names are feminine when they end with the letters *e or o*, and masculine otherwise, **except** for the following which are masculine:

- *le* Belize
- *le* Cambodge
- *le* Honduras
- *le* Mexique
- *le* Mozambique
- *le* Costa Rica
- *le* Zimbabwe

[1]Let's **write a function** named `toFrenchGender()` that takes the French name of a country and returns the name with the appropriate article added: le for masculine or *la* for feminine, such as *le* Canada or *la* Belgique. However, if the country name starts with a vowel, use *l'*; for example, *l'*Afghanistan. And, for another wrinkle, use *les* for plural country names. A name is plural if it ends in *es*, *is*, *as*, or *os*, or, if the first word in the name is *iles*, or "islands".

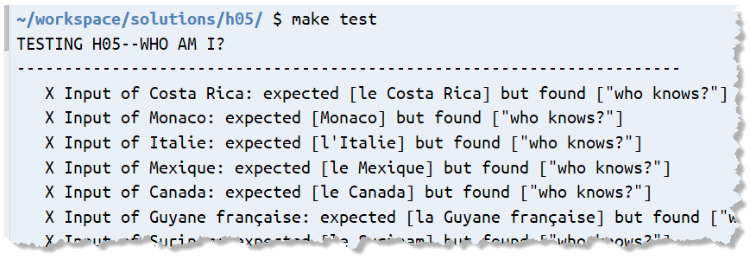
Finally, the following countries do not have a preceding definite article (*le* or *la*): Israel, Madagascar, Sri Lanka, Singapore, Monaco, Cuba and Cyprus. Note, I have removed all of the accents on the characters to make things easier when searching.

[1] Problem from Cay Horstmann's Big Java, 6th Edition, Chapter 5

Open the `.cpp` and define the function where noted. These are the specifics:

- **Return type** is `string`
- Function **name**: ***`toFrenchGender`***
- **Parameter**: ***`country`***, which, as you know from your reading this chapter, should be a **constant reference** to a `string`.

Once you've written the header, add a **`result`** variable and set it to **`"who knows?"`** Return the **`result`** at the end of the function, and then type **`make test`**. Here's what the output looks like. The test program selects fifteen random countries from 100.



```
~/workspace/solutions/h05/ $ make test
TESTING H05--WHO AM I?
-----------------------------------------------------------------------
   X Input of Costa Rica: expected [le Costa Rica] but found ["who knows?"]
   X Input of Monaco: expected [Monaco] but found ["who knows?"]
   X Input of Italie: expected [l'Italie] but found ["who knows?"]
   X Input of Mexique: expected [le Mexique] but found ["who knows?"]
   X Input of Canada: expected [le Canada] but found ["who knows?"]
   X Input of Guyane française: expected [la Guyane française] but found ["w
   X Input of Surin[illegible]
```

If you're tempted to use brute force to solve the problem, checking each value and then writing a series of **`if-else`** statements, you'll get a different sequence each time the program runs. In addition, the test program checks to make sure you **don't have more than 75 lines of code**, to prevent that.

Step 1 – Planning out the Processing

We need a plan. Here's some pseudocode that should work. Add this as comments between creating the result and returning it, and then let's implement the plan.

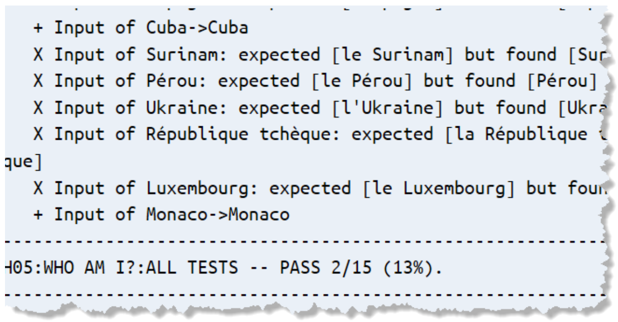
```
Country end in letter e or o? Feminine prefix->la (space)
Otherwise? Masculine prefix->le (space)
With these exceptions
   Belize Cambodge Mexique Mozambique Zaire Zimbabwe (all le)
   Starts with a vowel? prefix l' (no space)
   Plural? prefix is les (space)
   Israel, Madagascar, Sri Lanka, Singapore, Monaco,
   Cuba and Cyprus have no prefix.
```

To solve this, we'll need to write some **`if`** statements, since there are several possible outputs. Usually, you start with the exceptions.

Here's a plan:

- Create a **`string`** variable, **`prefix`**, inside the processing section
- At the **end of the processing section**, assign **`result`** the concatenation of **`prefix`** and **`country`** and return it.

Now, run your tests again. Depending upon what countries are picked, some of the tests will probably pass; which is encouraging, but doesn't really help very much.



```
   + Input of Cuba->Cuba
   X Input of Surinam: expected [le Surinam] but found [Sur
   X Input of Pérou: expected [le Pérou] but found [Pérou]
   X Input of Ukraine: expected [l'Ukraine] but found [Ukra
   X Input of République tchèque: expected [la République t
que]
   X Input of Luxembourg: expected [le Luxembourg] but foun
   + Input of Monaco->Monaco
------------------------------------------------------------
H05:WHO AM I?:ALL TESTS -- PASS 2/15 (13%).
------------------------------------------------------------
```

To solve the problem "for real", so **all** of the tests pass, we have to discover if a word starts with a vowel or not, or find out if it ends with a particular suffix.

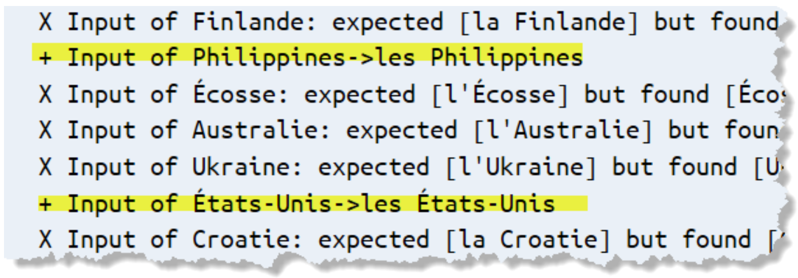
Step 2: Plurals

Let's start by handling the plurals. Implement this pseudocode:

```
Set variable islands to "iles"
Set variable len to the size of the string country
Set variable last to the last two characters in country

If any of these are true
        country starts with the value in islands. (use substr) OR
        last is one of "es", "is","os", or "as"
Then set prefix to "les ". (Note space)
```

Test again, (maybe a few times) and all of the plurals should be correct.



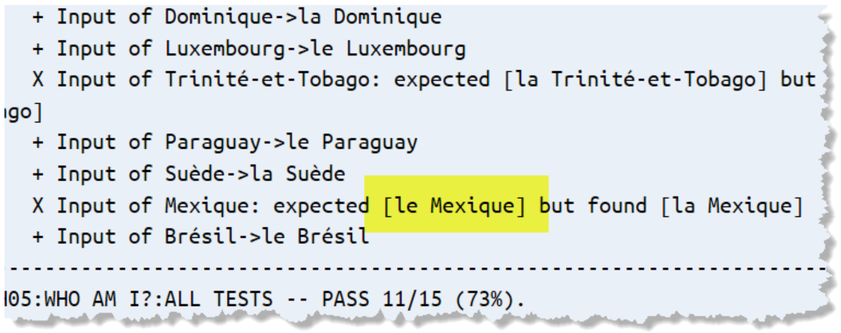
```
 X Input of Finlande: expected [la Finlande] but found
 + Input of Philippines->les Philippines
 X Input of Écosse: expected [l'Écosse] but found [Écos
 X Input of Australie: expected [l'Australie] but foun
 X Input of Ukraine: expected [l'Ukraine] but found [U
 + Input of États-Unis->les États-Unis
 X Input of Croatie: expected [la Croatie] but found ['
```

Step 3: Ladies & Gentlemen

Next, let's handle the feminine and masculine names. Remember, a French feminine name ends in an 'e'. This should be pretty straight-forward.

```
Elseif last character is an 'e' or an 'o' then
    Set prefix to "la "
Else
    Set prefix to "le "
```

However, if we try this out, we'll find that **some countries don't work correctly**. To fix that, we'll have to check **if the country begins with a vowel**, or, if is one of the special-case countries such as Mexico.

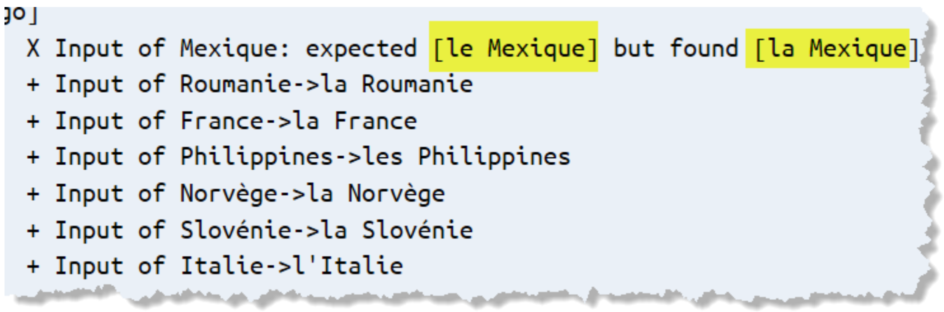


```
  + Input of Dominique->la Dominique
  + Input of Luxembourg->le Luxembourg
  X Input of Trinité-et-Tobago: expected [la Trinité-et-Tobago] but
ıgo]
  + Input of Paraguay->le Paraguay
  + Input of Suède->la Suède
  X Input of Mexique: expected [le Mexique] but found [la Mexique]
  + Input of Brésil->le Brésil
---------------------------------------------------------------
ı05:WHO AM I?:ALL TESTS -- PASS 11/15 (73%).
```

Use the `find()` member function to check for vowels. Here's some pseudocode.

```
Set variable vowels to "AEIOU" (only care about first letter)
Set variable first to the first letter in country
If first cannot be found in vowels
   Then set prefix to "l'". (no space)
```

Test again, and a few more should be correct:



```
go]
 X Input of Mexique: expected [le Mexique] but found [la Mexique]
 + Input of Roumanie->la Roumanie
 + Input of France->la France
 + Input of Philippines->les Philippines
 + Input of Norvège->la Norvège
 + Input of Slovénie->la Slovénie
 + Input of Italie->l'Italie
```

Step 4 - Special Cases

We have two sets of special cases.

- Countries that end in *e*, but are, none-the-less masculine, such as *Mexique*.
- Countries that don't have any prefix at all.

Using sequential if statements, make sure that **these exceptions go first**, otherwise they'll never be seen. Here's the pseudocode to implement:

```
Set variable plain to "Israel, Madagascar,…" (include all)
Set variable masculine to "Belize, Cambodge, Mexique, Mozambique …"
If country can be found in masculine
   Then set prefix to "le ". (space)
Elseif country can be found in plain
    Then set prefix to ""
Elseif – continue on with the original code
```

Go ahead and run `make test` and you can turn in the assignment with `make submit`.

Your Own Testing

Sometimes, you'd like to just **call the function** and see what happens. You can do that by placing any code you like inside the `run()` function at the bottom of the `.cpp` file. To run this code, use `make run` or `make stest`. Both will do the same thing.