FICHA FORMATIVA 7.1 : EXERCÍCIOS EXTRA- [AF7.1] - 08/04/2022

Clone emptyScala3Project (https://bitbucket.org/upskills-isep/emptyscala3project.git) and open it in Intellij IDEA.

Create the asked functions inside a scala object and write scalatest tests to test them.

1.EXTRA EXERCISES

It is possible to specify that a function has an undefined number of parameters using the symbol `*` as shown in the following example:

```
def tapAverage(tests:Double*)(assignments: Double*)(quizzes:Double*)=
       0.45*tests.sum/tests.length + 0.45*assignments.sum/assignments.length
+0.1*quizzes.sum/quizzes.length

//> tapAverage: (tests: Double*)(assignments: Double*)(quizzes: Double*)Double

tapAverage(18.7,16.3)(11.5,12.1)(18.1,19.9)      //> res0: Double = 15.085
```

Define functions `combineLists` and `combineLists2` that combine an arbitrary number of arguments of type `List[Double]` by using a certain operation, but not in the same way:

```
combineLists(List(18.7,16.3), List(11.5,12.1), List(18.1,19.9))("sum")
//> res1: List[Double] = List(48.3, 48.3)

combineLists2(List(18.7,16.3), List(11.5,12.1), List(18.1,19.9))("sum")
//> res2: List[Double] = List(35.0, 23.6, 38.0)
```

Write an implementation of each of the functions `combineLists` and `combineLists2` considering the templates:

```
def combineLists(ls:List[Double]*) (op: String)=
       ???


def combineLists2(ls:List[Double]*) (op: String)=
       ???
```

The functions can deal only with some predetermined operations, but at least four are to be considered: `sum`, `product`, `min` and `max`.