# Basic Exercises Part 9.1. Objective-C. Introduction

## Objective - C (Introduction and Exercises)

* Objective-C is a thin layer on top of C, and is a "strict superset" of C, meaning that it is possible to compile any C program with an Objective-C compiler, and to freely include C language code within an Objective-C class. Objective-C derives its object syntax from Smalltalk. All of the syntax for non-object-oriented operations (including primitive variables, pre-processing, expressions, function declarations, and function calls) is identical to that of C, while the syntax for object-oriented features is an implementation of Smalltalk-style messaging
* For more information about the language and history, visit:   
  <https://en.wikipedia.org/wiki/Objective-C>  
  And in the official Apple’s page:  
  <https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40011210>

### **1.1 Implement all the next exercises.**

### Create a basic Single View. We will create a new project on each approach, so we only going to write the steps once.

### **1.2** **Hello World!**

The objectives are simple:

* Write a function that returns the string "Hello, World!".

Inside the .h file

#import <Foundation/Foundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

**@interface** HelloWorld : NSObject

- (NSString \*)hello:(NSString \*)input;

**@end**

NS\_ASSUME\_NONNULL\_END

Inside the m file

#import "HelloWorld.h"

**@implementation** HelloWorld

- (NSString \*)hello:(NSString \*)input {

NSString \*message = @"Hello, %@!";

**if** (!input) {

input = @"World";

}

**return** [message stringByReplacingOccurrencesOfString:@"%@" withString:input];

// return [NSString stringWithFormat:@"Hello, %@!", input];

}

**@end**

### **1.3 One for X, one for me.**

### If the given name is "Alice", the result should be "One for Alice, one for me." If no name is given, the result should be "One for you, one for me."

### **1.4 Baby Bob**

Bob answers 'Sure.' if you ask him a question.

He answers 'Whoa, chill out!' if you yell at him.

He says 'Fine. Be that way!' if you address him without actually saying anything.

He answers 'Whatever.' to anything else.

### **1.5 Leap**

Given a year, report if it is a leap year.

The tricky thing here is that a leap year in the Gregorian calendar occurs:

on every year that is evenly divisible by 4

except every year that is evenly divisible by 100

unless the year is also evenly divisible by 400

For example, 1997 is not a leap year, but 1996 is. 1900 is not a leap year, but 2000 is.

If your language provides a method in the standard library that does this look-up, pretend it doesn't exist and implement it yourself.

### **1.6 Gigasecond**

Calculate the moment when someone has lived for 10^9 seconds.

A gigasecond is 10^9 (1,000,000,000) seconds.

### **1.7 Difference of Squares.**

Find the difference between the square of the sum and the sum of the squares of the first N natural numbers.

The square of the sum of the first ten natural numbers is (1 + 2 + ... + 10)² = 55² = 3025.

The sum of the squares of the first ten natural numbers is 1² + 2² + ... + 10² = 385.

Hence the difference between the square of the sum of the first ten natural numbers and the sum of the squares of the first ten natural numbers is 3025 - 385 = 2640.

### **1.8 Reverse string**

Reverse a string.

For example: input: "cool" output: "looc"

### **1.9 Sum of multiples**

Given a number, find the sum of all the unique multiples of particular numbers up to but not including that number.

If we list all the natural numbers below 20 that are multiples of 3 or 5, we get 3, 5, 6, 9, 10, 12, 15, and 18.

The sum of these multiples is 78.

### **1.10. The Hamming distance**

### The 'Hamming distance'. It is found by comparing two DNA strands and counting how many of the nucleotides are different from their equivalent in the other string.

GAGCCTACTAACGGGAT

CATCGTAATGACGGCCT

^ ^ ^ ^ ^ ^^

The Hamming distance between these two DNA strands is 7. Find The hamming distance given the two strings shown above.

### **1.11 Beer song**

Recite the lyrics to that beloved classic, that field-trip favorite: 99 Bottles of Beer on the Wall.

Note that not all verses are identical.

99 bottles of beer on the wall, 99 bottles of beer.

Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.

Take one down and pass it around, 97 bottles of beer on the wall.

……

2 bottles of beer on the wall, 2 bottles of beer.

Take one down and pass it around, 1 bottle of beer on the wall.

1 bottle of beer on the wall, 1 bottle of beer.

Take it down and pass it around, no more bottles of beer on the wall.

No more bottles of beer on the wall, no more bottles of beer.

Go to the store and buy some more, 99 bottles of beer on the wall.

* Optimize for readability, even if it means introducing duplication.
* If you've removed all the duplication, do you have a lot of conditionals? Try replacing the conditionals with polymorphism, if it applies in this language. How readable is it?

### **1.12 Acronym**

Write a program that converts a long name like “Portable Network Graphics” to its acronym “PNG”.

### **1.13 Prime number**

Given a number n, determine what the nth prime is.

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13.

If your language provides methods in the standard library to deal with prime numbers, pretend they don't exist and implement them yourself.