Lab 2: Objective-C

Due Date
11:59 PM, Monday September 21st 2015
Email zipfile “LastName-Lab2.zip” to cse436ta@gmail.com

Assignment – 10 points
This assignment will require you to write small portions of Objective C. It is designed to make you become more familiar with the language, as well as learn about the help features available. Don’t be alarmed when you don’t know how to program what you want in Objective C. The help tools and automatic fill in will come in handy.

The Objective-C runtime provides a great deal of functionality, and the gcc compiler understands and compiles Objective-C syntax. The Objective-C language itself is a small set of powerful syntax additions to the standard C environment.

To that end, for this assignment, you’ll be working in the most basic C environment available – the main() function.

This assignment is divided up into several small sections. Each section is a mini-exploration of a number of Objective-C classes and language features. Please put each section’s code in a separate C function. The main() function should call each of the section functions in order.

The basic layout of your program should look something like this:

```c
#import <Foundation/Foundation.h>

// sample function for one section, use a similar function per section
void PrintMDict() { // Code from path info section here
}

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        PrintMDict(); // Section 1
        PrintSet(); // Section 2
        PrintDate(); // Section 3
        PrintRNumb(); // Section 4
    }
    return 0;
}
```
Grading
This assignment is different in that when grading, we generally only care about the output. Here, we will also grade based on the code provided for each of the four sections. As in the previous assignment, points will be deducted for each warning. If you run into warnings you don’t know how to fix, simply Google the error messages. Most of the time the fix will be require adding nothing more than a couple of words.

Output will be printed to console using NSLog. You should follow the directions of each section, which will have you print out variables using NSLog. Hard coding the NSLog statements will earn no points.

A note about the NSLog function. It will generate a string that is prefixed with a timestamp, the name of the process, and the pid. It will also automatically append a newline to the log statement.

2009-09-09 13:49:42.275 LastNameLab2[360] Your message here.

This is expected. You are only being graded on the text not generated automatically by NSLog. You do not need to attempt to suppress what NSLog prints by default. To help make the output of your program more readable, it would be helpful to put some kind of header or separator between each of the sections (such as a line of asterisks *****).

You should use the lecture notes, Apple resources from the Apple Developer website


and also the choices found under the Help menu option inside of Xcode.

QUICK TIP: Remember that an Objective-C NSString constant is prefixed with an @ sign. For example, @"Hello World". The compiler will warn, and your program will crash if you use a C string where an NSString is required.

To start off with, open a new project. Under the OS X option, choose Application, and then choose Command Line Tool. For product name, it helps grading if your project is named LastNameLab2.
You will be doing the bulk of your work in the main.m file where a skeletal implementation of main() has been provided. We recommend that you go through and complete one section at a time, testing it to make sure there are no errors or warnings. The sections will progress in difficulty!

**Section 1 – A Measurable Dictionary (PrintMDict() )**

In this section, you will create a small mutable dictionary that will keep track of different measurements, as well as what type of measurement is measured. (We will use a shortcut and associate in our heads a type of measurement with a number. Metric Length will be 1, Metric Weight will be 2, Standard Length will be 3, Standard Weight will be 4). The key will thus be an NSString, while the value will be an int.

Add the following key/value pairs to the dictionary.

- Centimeter 1
- Pound 4
- Ounce 4
- Kilogram 2
- Yard 3
- Millimeter 1
- Kilometer 1

Enumerate through the keys of the dictionary, checking to see if each string ends with @"meter". If it does, print out the value associated with the key (the int using NSLog). While it is obvious that the answer should be a 1 printed out 3 times, you will receive no credit if you hardcode this. i.e. Do not type NSLog(@"1") three
Hints
NSString has methods to determine if a string has a particular prefix or suffix. When using NSLog, remember that an int is not an NSString and thus should not have an @ before the type of variable. (What type of type is associated with an int? - https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Strings/Articles/formatSpecifiers.html)

Section 2 – Sets and Sets (PrintSet() )
Create two NSMutablesSets of int’s. Add the following seven elements to the first set: 5, 3, 2, 9, 2, 9, 7. Add the following five elements to the second set: 3, 9, 2, 7, 7. By using functions on the set, answer the following questions: (Print out the answer in the form of NSLog with one answer per line. Include the question number on each line (with the answer following). )
1) Is the first set a subset on the second set? (i.e. is Set1 C Set 2?)
2) Is the second set a subset of the first set?
3) Is 4 an element of the first set?
4) Is 2 an element of the second set?
5) Does one set intersect the other set?

Hints
Remember that you must use already defined functions to answer each question. For example, if one of the questions were how big is set 1, you could not use a for loop to iterate through the set and print out the loop variable. Similarly, as in other sections you cannot simply type out an answer. 2 is clearly an element of the second set, but you may not hardcode the answer. Finally, remember to output the question number on the same line as the answer!

Section 3 – What day is it? (PrintDate() )
NSDate is a necessary data type when dealing with dates. Your task is to create several NSDate variables, based on the specifications listed below. You must then find a way to format each date using NSDateFormatter, and print out the result using NSLog. That is, the output should be something completely readable and not just something like NSLog('%@', date1).

IMPORTANT: The dates reported should be the date the project is run by the TA. You will not know this date (and it is also not the date the lab is due). DO NOT use the date you are writing this section on.

1) Today's date (i.e. the date the lab is tested by the TA)
2) The date 8 days ago (i.e. 8 days before date 1, or 8 days before the day the lab is being tested on)
3) Last Tuesday's date
4) The earlier date between: 8/27/12 and 9/13/12 (let Xcode determine which one is the earlier date)
5) The later date between: 1/10/12 and 12/20/11 (again, let Xcode determine this)

**Hints**
You may need to make some NSDate variables that are not specifically being asked for. That is perfectly fine; just print out the five that you are specifically being asked for. This section should be really easy if you read through Apple’s documentation! There are Xcode functions to do each of the above in no more than ten lines total. Please also make sure you convert the date to a readable format as specified in the introduction to this section.

**Section 4 – Random Numbers (PrintRNumb() )**
Have Xcode generate five random (well pseudorandom ...) numbers between 0 and 9. There are several ways to have Xcode generate the numbers (the distribution should be uniform); just don’t type in random numbers yourself. Create an NSMutableArray of NSString’s adding in each number's text equivalent (i.e. 1 should become “One”, 7 should become “Seven”, etc.) Iterate through the array and print out the text values of each number – one number per line (for five lines total).

**Hints**
This should be fairly straightforward. You can look up how to generate random numbers and find plenty of results and examples.

**Grading**
The usual automatic deductions apply.

Each section is worth 2.5 points and graded separately. You will lose points for not following directions.