

CS 646 iPad/iPhone Mobile Application Development
Fall Semester, 2014
Syllabus
© 2014, All Rights Reserved, SDSU & Roger Whitney
San Diego State University -- This page last updated 8/22/14

CS 646 Syllabus

Instructor	Roger Whitney
Office	GMCS 561
Phone	594-3535
Email	whitney@sdsu.edu
Office Hours	3:15-5:15 pm Tuesday, Thursday, 10 am - noon Friday

Course Objectives: Learn the basics of developing iOS apps.

Course WWW Site: The course website is in the SDSU Blackboard system. SDSU students can access it through their SDSU portal. It can also be access at: <https://blackboard.sdsu.edu/>

Texts:

- Objective-C Programming: The Big Nerd Ranch Guide (2nd Edition), Aaron Hillegass, Mikey Ward 2013, Addison-Wesley Professional, 978-0321942067
- Beginning iOS 7 Development: Exploring the iOS SDK Jack Nutting, Fredrik Olsson, David Mark, Jeff LaMarche, Apress, 2014, 978-1430260226
- Programming iOS 6, Third Edition edition, Matt Neuburg, O'Reilly Media, March 21, 2013, 1449365760 (Optional)

The mobile development is changing rapidly. Each year Apple releases a new version of iOS and updated versions of their development tools. As a result books tend to be slightly out of date when they are published. For the most part this does not matter as the basics remain the same. You do need to pay attention to the version of Xcode you are using and the version of Xcode discussed in the textbooks. If the versions are different beware that some of menu items may be different between the versions of Xcode. Sometime during the semester Apple is making major changes to iOS and Xcode. This will impact the course. The new version of Xcode has some important improvements. One of those improvements this year is the new language Swift.

Prerequisites: The course assumes that you have some programming experience in a C based programming language (Java, C, C++ or C#) and that you have some experience working with APIs in a C based programming language. You will be developing applications for iPhone/iPod touch/iPads. This involves designing and building GUIs and using SQL databases. While prior experience with GUIs and SQL databases are not required prior experience will be very useful in the course.

No prior knowledge of Objective-C, iOS development or mobile development is needed.

Required Hardware: The recommended hardware/software for the course is a Macintosh computer running Mountain Lion (Mac OS 10.8.4). These are required to run the current version of Apple's tools for developing iPhone applications. During the semester Apple will release Xcode 6 which requires Mac OS 10.9 (Mavericks). For on campus students that do not have Macintosh computers the department has a lab with Macs for this course. Off campus students need access to a Macintosh computer preferably running Mountain Lion (or later) in order to take this course. An iOS device (iPhone, iPad, iPod Touch) for testing your apps is not required but is strongly recommended.

Apple Developer Program: Apple has a developer program for iOS developers which costs \$100 per year. You do not need to sign up as a developer with Apple for this course. The university is an Apple University developer, which give students free access to the Apple developer program. You will not be able to submit apps to the App store using the university developer program.

Grading: Your grade will be based on one exams (20% of your grade) assignments (60% of your grade), a project (20%). The course will be graded on 1,000 points. The exams will be on Sept 18. There will be 6 assignments. Tentative dues dates for the assignments are Sept 4 , Sept 14, Sept 28, Oct 12, Oct 26, and Nov 11. The project is due Dec 11.

Exams & On-line Students: There will be one exam in the course. The on-line students need to find someone to proctor the exam for them. Local libraries and local colleges often have a service that proctors exam for on-line students. Also religious organizations often will also proctor exams. Once you have identified who will proctor your exam please send their contact information to the instructor (Roger Whitney).

Dropping the course. The following is from the CES website from <http://www.ces.sdsu.edu/Pages/Engine.aspx?id=64#refund>. If you drop this course CES will refund part of your fees depending on when you drop the course. If you drop the course at least one day before the first time the class meets you will receive a full refund minus \$17 administration fee. If you drop the course after the first meeting of the class but before 25% of the course time as elapsed you will receive 65% of your fees back minus \$17 administration fee. After this date you will not receive refund if you drop the course. If you wish to drop this course to take another of the Certificate course this semester you can do so without any penalty. If you have questions about this and other issues related to CES please contact the CES registration office at 619-594-5152.

Late Policy: Late homework will be accepted, but with a penalty. An assignment turned in 1-7 days late, will lose 3% of the total value of the assignment per day late. The eight day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.

Crash Policy: The last day to add this course is Sept 8.

Cheating: Any one caught cheating will receive an F in the course and they will be reported to the SDSU Judicial Procedures Office.

Course Project: Due December 11. There is a course project required for this course. You can work on the the course project by your self or in teams of two people. Teams larger than 2 are not allowed. You are to come up with an idea for an iPhone or iPad application and implement the application. Your project will be evaluated using the following criteria:

1. Originality (5 points)
How original is the idea and/or implementation of the project.
2. Size of project (20 points)
Is the project an appropriate size for a semester project. A project that is too large is as bad as a project that is too small. The project should be larger in scope than an assignments in the course? The project should take longer than 2-3 week to design and implement?
3. Quality of UI (80 points)
The app should follow the Apple UI guidelines for iOS apps. How well does the app follow users mental model of how the app should work. Are the UI elements used effectively or not? Is the app UI structured in a way to make it easy to use and understand. Is the text used in menus, labels, buttons, etc. concise and have clear meaning?
4. Working code (75 points)
Do the features implemented work. All UI elements should actually do what they are supposed to do? Are features fully or only partially implemented. Does the app have enough features to actually do something. Does the app run? Are there bugs and memory leaks? Does the app crash?
5. Quality of code (20 points)
The code should be formatted in a reasonable and consistent manner. Names of classes, methods and variables should understandable and follow standard naming conventions. The code should be clear and well organized. The code should be appropriately documented.

Crash Policy: The last day to add this course is Sept 9.

Cheating: Any one caught cheating will receive an F in the course and they will be reported to the SDSU Judicial Procedures Office.

Disabled Students: If you are a student with a disability and believe you will need accommodations for this class, it is your responsibility to contact Student Disability Services at (619) 594-6473. To avoid any delay in the receipt of your accommodations, you should contact Student Disability Services as soon as possible. Please note that accommodations are not retroactive, and that accommodations based upon disability cannot be provided until you have presented your instructor with an accommodation letter from Student Disability Services.

Topics covered in the course:

Objective-C

Objective-C is a super set of C. That is any C program is a legal Objective-C program. The course assumes that you know a C based language. So the basic C syntax (assignments, functions, loops, etc) will not be covered in this course. The parts of Objective-C that are not in C will be covered. This includes classes, message syntax, protocols, properties, memory management (reference counting and automatic reference counting), blocks and foundation classes.

Xcode 5 & 6

Xcode 5 (and 6) will be used in the course. The course will spend a little time looking at Xcode 5 IDE and the parts needed to develop apps - the editor, debugger, UI builder and Instruments. We will switch to Xcode 5 when it comes out.

UI design

Development teams for mobile applications tend to be small. So developers are often involved in UI design. Some background in UI design will be covered - affordance, conceptual models, feedback, information structure. Parts of the iOS Human Interface Guidelines will be examined.

iOS API

The vast majority of the course will be spent covering the basics of the iOS API needed to develop apps. Topics listed below correspond to chapters 1-17 in Beginning iPhone 7 Development. Topics will be covered in the order listed.

- Model-View-Controller

- Connecting views and controllers - xib files, outlets, actions

- Application delegate

- Basic views and controllers

- Text fields, sliders, switches, buttons, segmented controls, action sheets, alerts

- Autorotation, autosizing

- Testing

- Multiview applications

- Tab bars and pickers

- Table Views

- Navigation controllers

- iPad views - split views and popovers

- Application settings

- Data Persistence

- Files, Property lists, SQLite3, Core Data

- Background Processing (Grand Central Dispatch)

- Drawing

- Taps, Touches, Gestures

- Location

- Motion

- Localization

- Swift

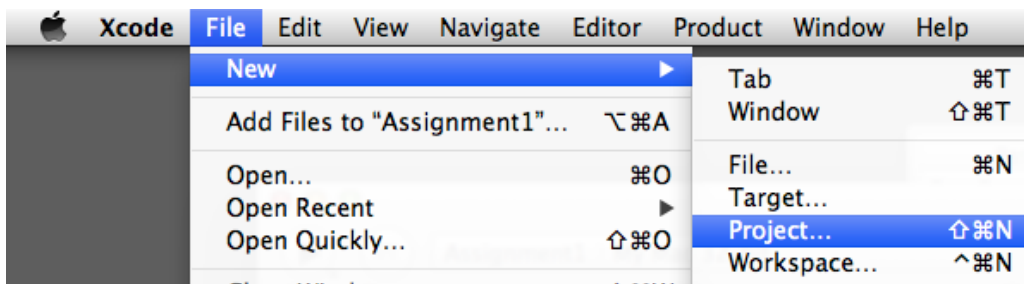
CS 646 iPad/iPhone Application Development
Fall Semester, 2014
Assignment 1
© 2014, All Rights Reserved, SDSU & Roger Whitney
San Diego State University -- This page last updated 8/25/14

Assignment 1 (100 points)
Due Sept 4 11:59 pm
Objectives

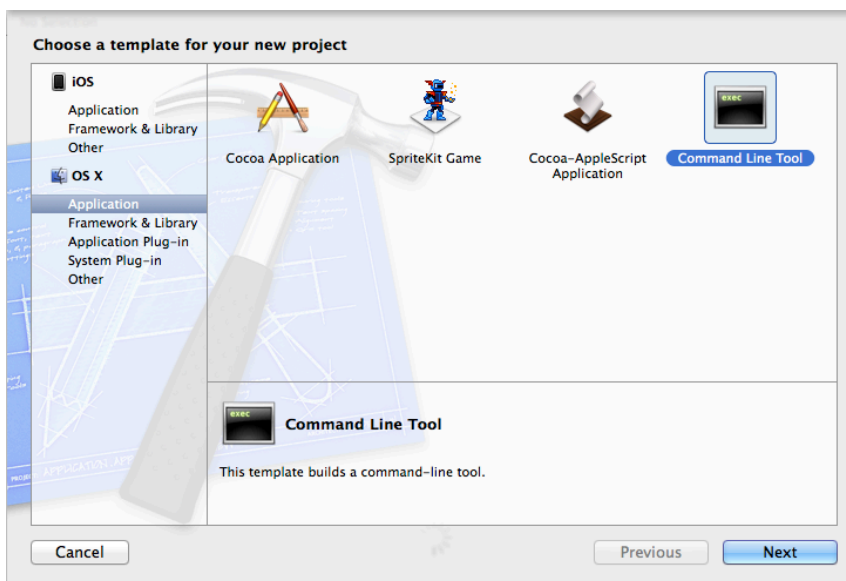
1. Get started using Xcode.
2. Learn basic Xcode operations.
3. Learn how to use Xcode documentation for iOS API.

Programing Problem (10 points each)

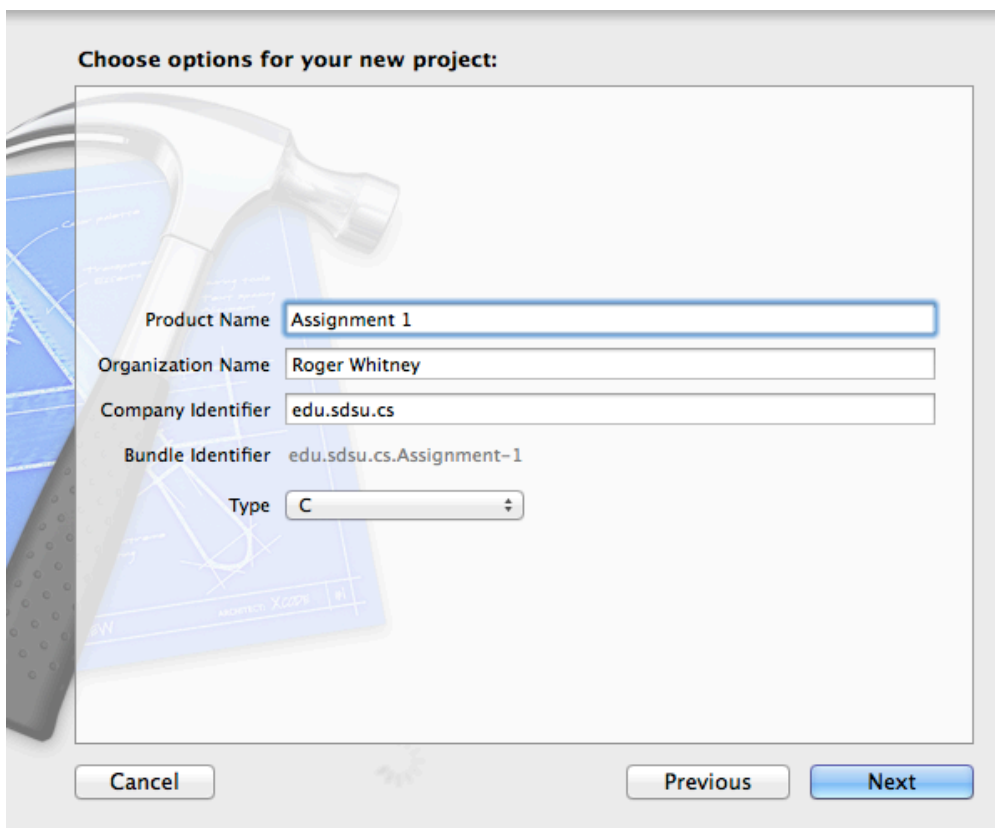
One of the main goals of this assignment to start getting familiar with Xcode. So you must use Xcode for this assignment. You will be turning in an Xcode project. In Xcode create a new project. To do this in the "File" menu select "New" then Project".



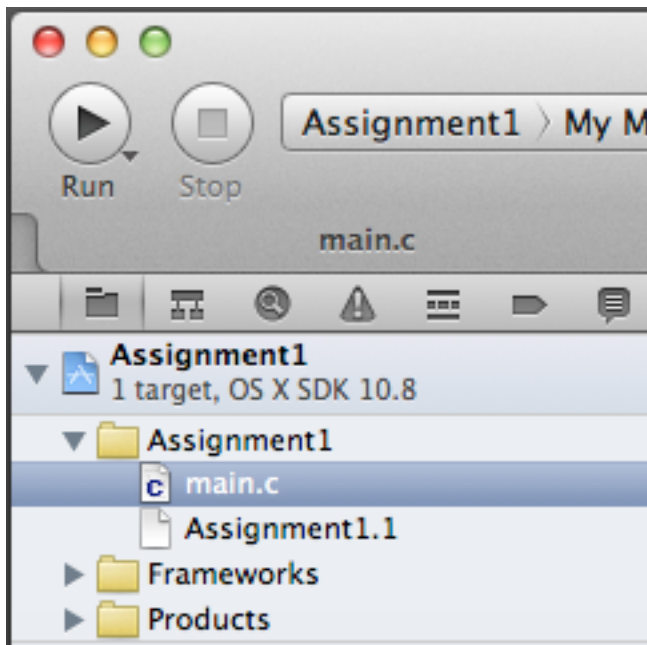
Then under OS X select "Application" then "Command Line Tool". Click on the "Next" button.



Fill in the next window with your information.



After clicking "Next" save your project. Once the project is created find the "main.c" file in the left most pane in Xcode. Click on the file to open it. Add your code to that file.



Now the programming questions.

1. Write a c function, call it polyTable, that has one argument, an integer. Lets call it N. The function prints out on the console the values k and $k^3 + 2*k - 3$ for the values $k = 1, 2, \dots, N$. Print each value of k on a separate line.

2. Add a breakpoint in the polyTable method. Step through the polyTable method until $k = 3$. In the debugger change the value of N to -1. What happens when you continue to run the program?
3. Write a c function, called runningSum, that has one int argument and returns an int. The return value is the sum of the current and past values of the arguments passed to the function. The history of the arguments restarts at zero each time your program starts. Test your function by calling it several times in main and printing the return value each time. Do not use global variables (think static).

```
//Start of program
x1 = runningSum(2);
// x1 == 2
x2 = runningSum(2);
// x2 == 4
x3 = runningSum(3);
// x3 == 7
x4 = runningSum(5);
// x4 == 12
```

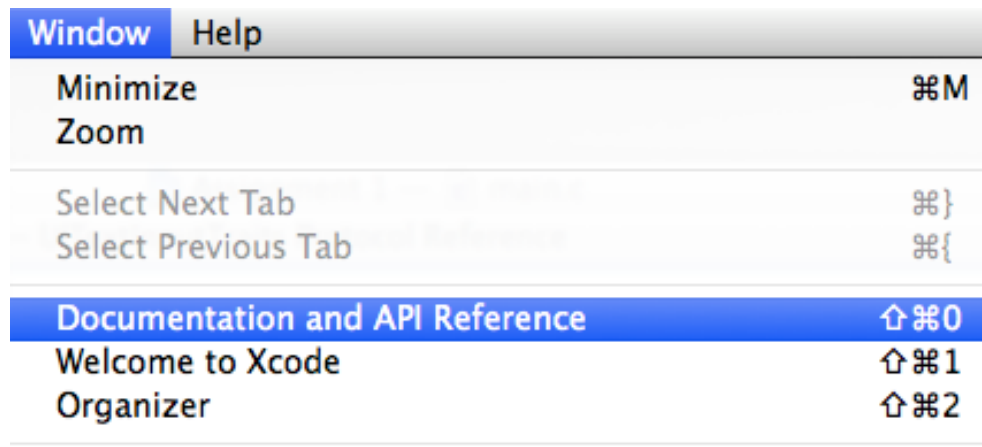
4. Add a break point at the start of your runningSum program. Have the break point trigger for the first time the third time runningSum is called. Change the value of your static variable to 2 then continue running the program. When you do this what will be the value of x_4 in the above code?
5. The Fibonacci numbers are defined as: $F_n = F_{n-1} + F_{n-2}$ where $F_1 = F_2 = 1$ and n is a positive integer. Write a recursive c function, called fibonacci, that with input n returns the n th Fibonacci number F_n .
6. Profile compute(1100) where compute is given below. Use the Time Profiler. (Select "Profile" in the "Product" menu. Then in the dialog window select "CPU" under "Mac OS X", then in the right pane select "Time Profiler" and click on the "Profile" button. In the profile results what percentage of time did the sin function take? The printf function?

```
float compute(int n) {
    float y = 1.1;
    for (int k = 0; k < n; k++)
        for (int j = 0; j < n; j++) {
            y = sin(k*j + y);
            printf("%f", y);
        }
    return y;
}
```

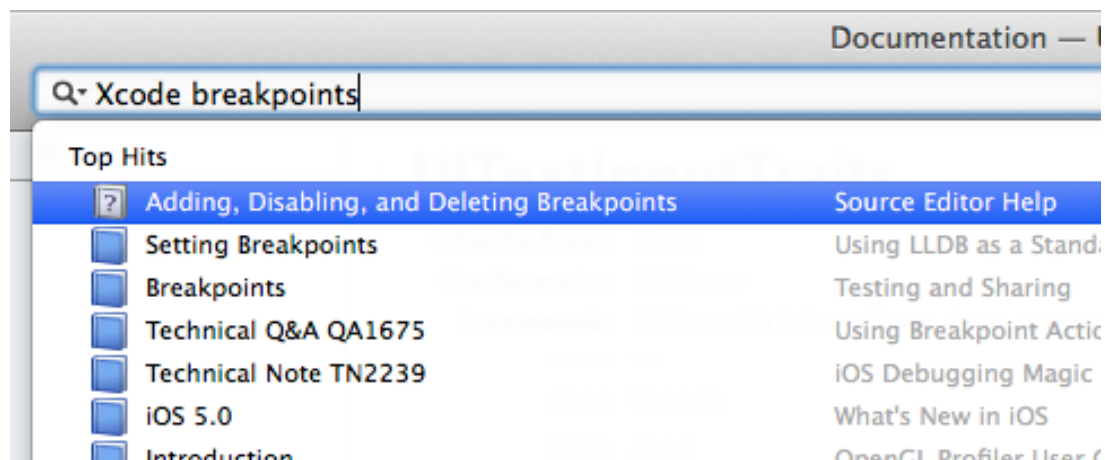
Questions (10 points each)

No doubt that you can find the answers to these questions using a Google search, but don't. The entire point of these questions is to get you used to using the documentation for iOS included with Xcode. So use the Documentation Viewer in Xcode to find the answers.

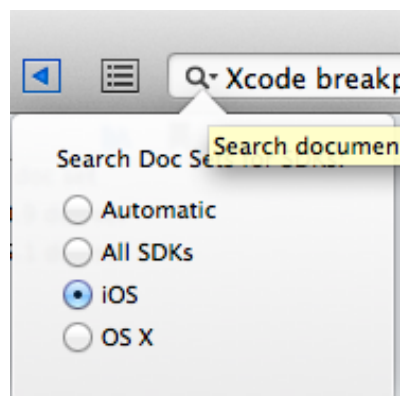
Once you have an Xcode project you can open the Documentation Viewer in Xcode selecting "Documentation and API Reference" in the "Window" menu of Xcode.



In the top pane of the Documentation Viewer there is a search box.



You can set the find options by clicking on the magnifying glass.



If you select "iOS " you can avoid reading about things that apply only to OS X development.

Now finally the questions.

7. What version of iOS added the method **description** to the class Object?

8. What method should one use instead of **+ stringWithContentsOfURL:** in NSString? Why is stringWithContentsOfURL: deprecated?
9. What is the difference between the methods **valueForKey:** and **objectForKey:** in the NSDictionary class?
10. What protocols does the NSArray class conform to?

What to Turn in

Create a Xcode project for the assignment. In your project add a file called "Answers.txt". In this file put the answers to the "Questions" 7-10 and "Programming Problems" 2, 4, 6. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. Turn in your zipped file using assignment 1 link on blackboard.

Late Penalty

An assignment turned in 1-7 days late, will lose 3% of the total value of the assignment per day late. The eighth day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.

Assignment 2
Due Sept 14 11:59 pm

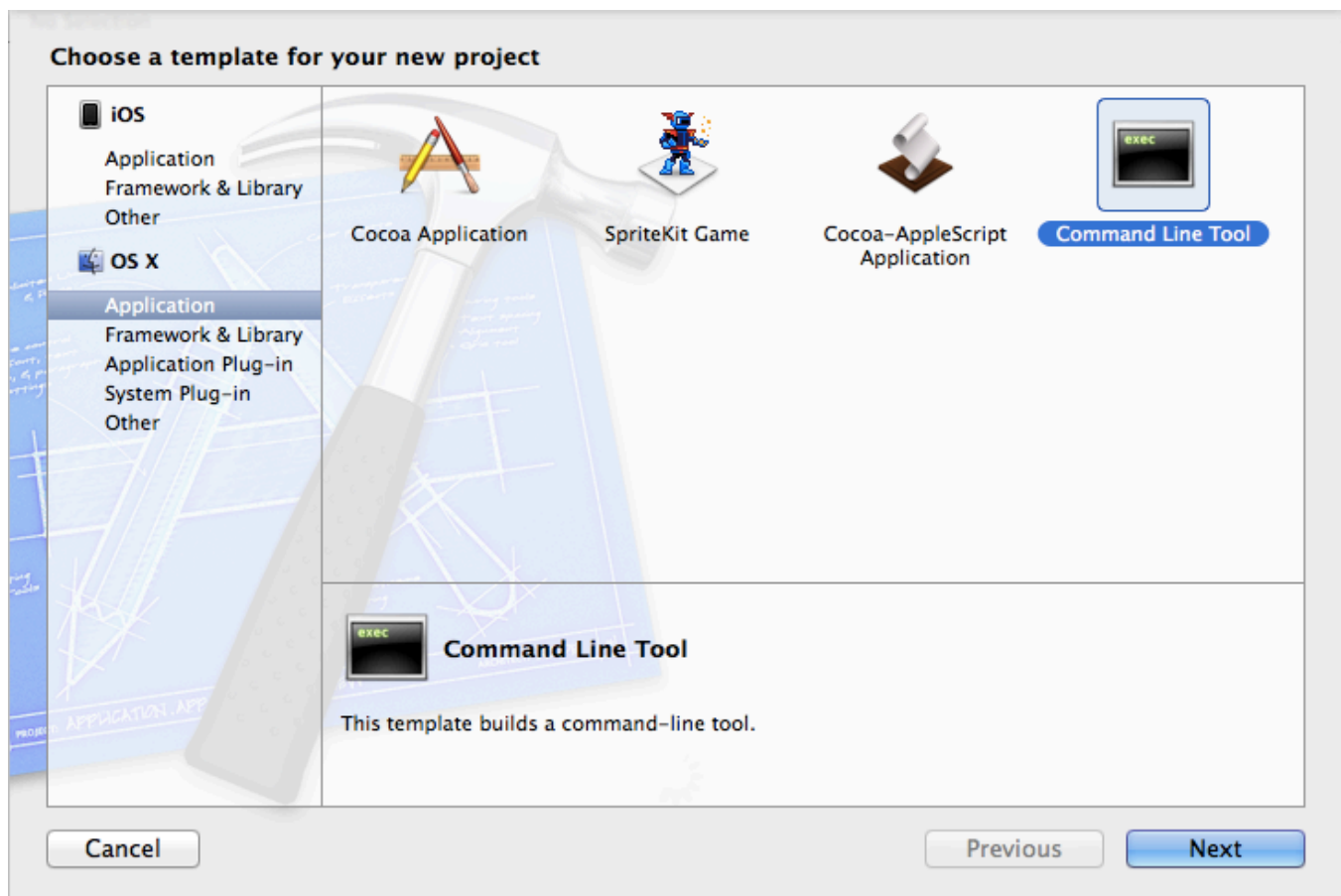
100 Points Total

Objectives

1. Use more Objective-C features - classes, protocol, extending classes, etc

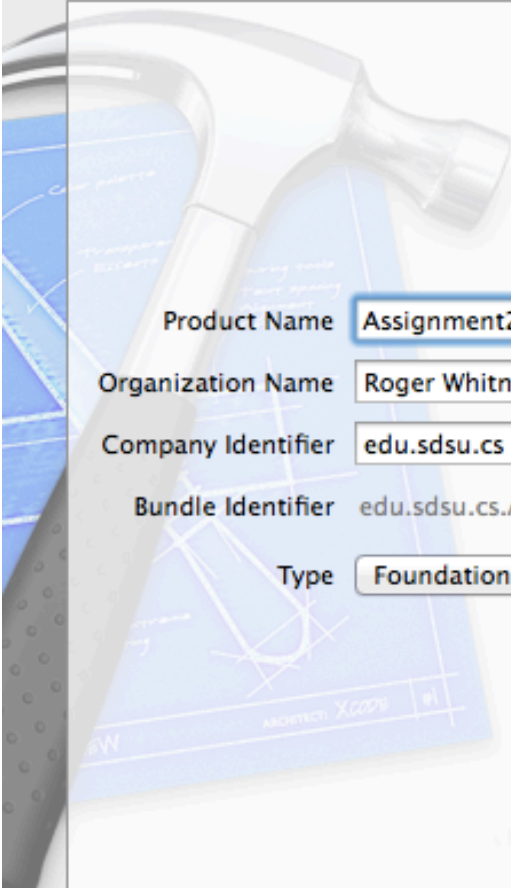
Programming Problems

When you create your Xcode project for this assignment select a command line tool.



And then select Foundation for the type.

Choose options for your new project:



Product Name

Organization Name

Company Identifier

Bundle Identifier

Type

1. (5 points) A phone number in this country has 10 digits like 6195946191 or 619JYDN191. It is hard to read a phone number formatted like that. One common format is 619-594-6191. Add the method **phoneFormat** to the NSString class. The method **phoneFormat** converts strings like @"6195946191", @"619 594 6191", @"619 5946191" and @"619-594-6191" to @"619-594-6191". That is all the methods below will return @"619-594-6191"

```
[@"6195946191" phoneFormat]
[@"619 594 6191" phoneFormat]
[@"619 5946191" phoneFormat]
[@"619-594-6191" phoneFormat]
```

2. (15 points) Create a PhoneNumber class. A PhoneNumber object has a number and a type. We will use only US phone numbers. A type can be mobile, home, work, main, home fax, work fax, pager or other. Defines static constants of type NSString for the types of phones. The phone class needs to have the following methods. It can have more methods if needed. Each class you create should use two files a header file (.h) and an implementation file (.m).

```
+ (id) type: (NSString *) phoneType number: (NSString *) number
```

Creates a new Phone object with given data

-(NSString *) number

Returns the phone number formatted as in problem 1.

-(BOOL) isMobile

Returns true if the phone number is for a mobile phone.

-(BOOL) isLocal

Returns true if the phone number has the area code 619 or 858.

-(NSString *) description

Returns a string description of the phone number in the format "type: number".
For example 619 594 6191 is a work number so it would be @"work: 619-594-6191"

3. (5 points) Add a method **asPhoneNumber** to the NSString class. The method takes a string of the format @"work: 619-594-6191" and returns a PhoneNumber object with the given types and number. If the string is not of the format "type: number" then the method **asPhoneNumber** throws an exception.

4. (15 points) Create a Name class. A name has two parts a first name (also called a given name or personal name) and a last name (family or surname). The Name class should have the following methods (more if needed).

+ (id) firstName: (NSString *) firstName lastName: (NSString *) lastName

Creates a new Name object with given data

-(NSString*) description

Returns full name of the person, (FirstName LastName)

-(NSComparisonResult)compare:(Name *) aName

Returns an NSComparisonResult indicating how the two names compare in order.

5. (25 points) Create a Person class. A Person has a first name, last (or family) name and 0 or more phone numbers. The person class should have the following methods (more if needed):

+ (id) firstName: (NSString *) firstName lastName: (NSString *) lastName

Creates a new Person object with given data

- setPhoneNumber: (PhoneNumber *) number

Adds a phone number.

-(NSString*) description

Returns full name of the person, (FirstName LastName)

-(NSString *) phoneNumber: (NSString *) phoneType

Returns the person's phone number of the given type. Or nil if number does not exist.

-(BOOL) hasNumber: (NSString *) phoneNumber

Returns true if person has given phoneNumber.

6. (25 points) Create a ContactList class. Internally the ContactList maintains one list of your contacts. Your class should have at least the following methods. You may find that you want/need more methods in the class.
- (void) addPerson: (Person *) newContact
Add a Person object to the list.
 - (NSArray *) orderedByName
Returns an NSArray of all your contacts ordered by last name.
 - (NSArray *) phoneNumberFor: (NSString *) lastName
Given the lastName return phone numbers for the first person in the list that has that last name. Return an empty array if no such person exists.
 - (NSString *) nameForNumber: (NSString *) phoneNumber
Return the full name of the person with the given phone number. Return nil if no one has the phone number.

Grading

In addition to the points indicated for each problem there will be 10 points for style. Style includes formatting your code reasonably and consistently, using Objective C naming conventions and using the appropriate language constructs.

What to Turn in

Create a Xcode project for the assignment. Make sure all the files you created are in the same project. You should have created at least 10 files. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. Turn in your zipped file using blackboard under assignment 2.

If you are using the SDSU library machines to do the assignment you will not be able to use the "Command line tool" project. Select "View-based Application" template under iOS Application. You can then modify the main method to call your code. When you run the project the simulator will run, but will not do anything.

Late Penalty

An assignment turned in 1-7 days late, will lose 3% of the total value of the assignment per day late. The eighth day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.

Assignment 3
Due TBD

Objectives

Introduction to iOS apps and related issues:

- Keyboard - hiding and selecting
- Dealing with text fields and buttons
- Saving data
- Using touch events to move objects on the screen (optional)

The Application

Implement a iOS app that contains three text fields (inputText, x, y) with labels, one button (with label Update) and one other label, for ease of explaining the assignment call this label the moving label. See the screenshots below.

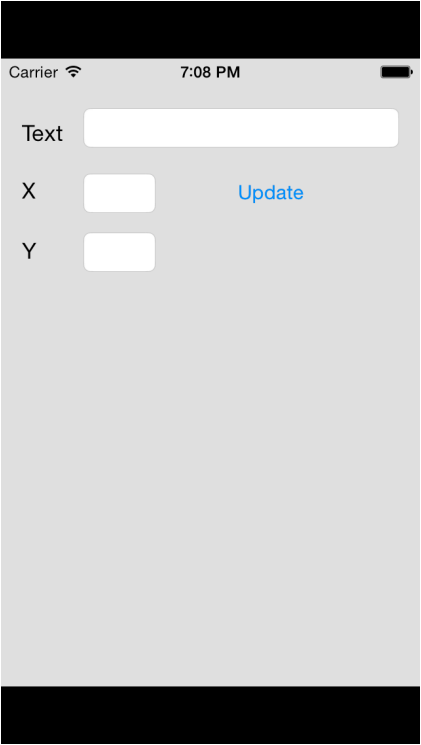
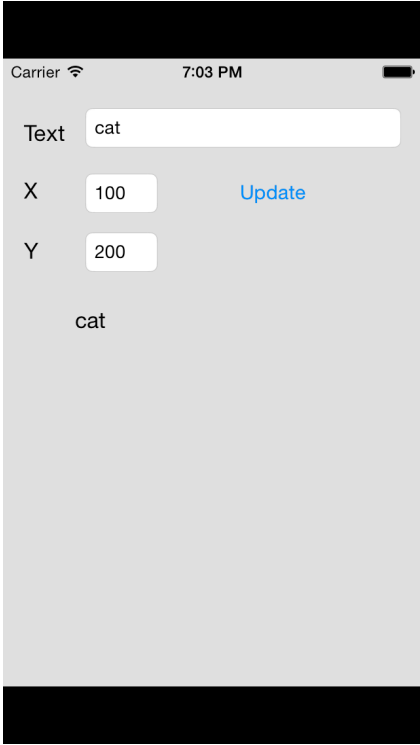
When the user clicks on the "Update" button with the value in the inputText field is copied to the moving label. If the x and y text fields have values then the center of the moving label is moved to that location (x, y). If either x or y text field do not have any text the moving label does not change position when the user clicks on the "Update" button. Clicking on the "Update" button hides the keyboard.

The application should store the values in the text fields. So when the user "kills" the application and then restarts the app the values in the text fields are restored to the values they had before the app was killed. Killing the app is different than just placing the app in the background.

If you are using Xcode 6 you should to turn off Auto Layout.

Optional

Allow the user to move the moving label by touching the screen with one finger. When the user touches the screen the center of the moving label is placed under the finger. When the moving label moves update the values in the text fields x and y.

Screen on First Startup	After Update
	

Grading

The assignment will be graded as follows:

Points	Item
10	View Contains button, textfields, labels
35	When the Update button is pressed the moving label is updated properly. You can assume that the user enters proper values.
10	Use appropriate keyboard
15	Keyboard is dismissed when the Update button is pressed
5	Design and look of interface, following Apple UI guidelines
20	App properly has correct values when restarted
5	Proper coding style

Points	Item
10	User touch moves the moving label (optional)

What to Turn in

Create a Xcode project for the assignment. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. Turn in the zip file.

Late Penalty

An assignment turned in 1-7 days late, will lose 3% of the total value of the assignment per day late. The eighth day late the penalty will be 40% of the assignment, the ninth day late the penalty will be 60%, after the ninth day late the penalty will be 90%. Once a solution to an assignment has been posted or discussed in class, the assignment will no longer be accepted. Late penalties are always rounded up to the next integer value.

Assignment 4 - Sampler App
Due Nov 2 23:59

Objectives

Sample some UI components.

App description

This app will use a number of UI elements so the app will not make much sense as a single app. You will use a tab bar with a number of tabs. The view for each tab is described below.

Food tab

The view at this tab will contain a custom picker view with two columns. The first column will contain a list of countries. When the user has selected a country then the second column displays a list of foods in that country. The list of countries and their foods are given below. The view also has a slider. When the user selects the top (first) food item in the list the slider moves all the way to the left. When the user selects the bottom (last) food item in the list the slider moves all the way to the right. When the user selects a food item between the first and last the slider moves proportionally. So if Mexico is selected and the user selects the second food item from the top then the slider moves 1/10 the way to the right. Conversely when the user moves the slider the picker will move to the corresponding food item. A slider has many possible positions. However the picker has much fewer positions. For example when Mexico is selected there are only 11 food items. So the picker has only 11 positions. So when the user moves the slider they may leaving in a position that does not correspond exactly to a food selection in the picker. When that happens after the user is done move the slider to the closest position corresponding to a food selection in the picker.

Country	Food Items
India	Avakaya, Pesarattu, Thukpa, Thali, Litti Chokha, Maple, Palak Paneer, Rajma-Shawl, Vindaloo, Khaman, Handva, Bisi bele bath, Pav Bhaji, Eromba, Chungdi Jhola
USA	Hot Dog, Pizza, Hamburger, Clam Chowder, Succotash, Fried Chicken, Gumbo, Grits, Chitterlings, Hushpuppies, Cobbler
Mexico	Taco, Quesadilla, Pambazo, tamal, huarache, Alambre, Enchilada, Panita, Gordita, Tlayuda, Sincronizada

Web Tab

This view contains a text field and a web view. When the user enters a valid url in the text field the web view displays the indicated web page.

Segment Tab

This view has a segmented control with three options: Progress, Text, Alert. When the user selects the "Progress" option they see a switch and an inactive activity indicator. When they turn the switch "on" or to the left the activity indicator spins. When they turn the switch off the activity indicator stops spinning. When the user selects the "Text" option they see a Text View that they can enter text. The text view should start with some text. When they select the "Alert" option they will see a button. When they click on the button an alert will pop up asking the user "Do you like the iPhone".

Grading

The assignment will be graded as follows:

Points	Item
10	Tab bar working
15	Picker in Food Tab working correctly
15	Slider in Food Tab working with Picker
10	Web view in Web tab working
10	Segment Control working in Segment Tab

Points	Item
10	Progress activity working in Segment Tab
10	Text view in Segment Tab
10	Alert in Segment Tab
10	Proper coding style

What to Turn in

Create a Xcode project for the assignment. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. This assignment we will not use blackboard to turn in assignments. We will use my older course portal (<http://bismarck.sdsu.edu/CoursePortal>).

Assignment 5 - Rate The Instructor
Due Nov 20 23:59

Objectives

Table Views, Navigation, network connections

App description

You will build an iPhone app that accesses information about instructors and lets people rate them. Use a table view to present some identifying information about each instructor. When the user selects an instructor they get a view of more detailed information about that instructor. There also is a way to rate an instructor on a scale of 1 to 5 (integer values only) and provide comments about an instructor. The information available about an instructor is:

First name
Last name
Office
Phone
Email
Rating
Comments

The information is available in JSON format using a REST-like interface via http using the following urls.

GET urls

<http://bismarck.sdsu.edu/rateme/list>

Returns an json array. Each element of the array is a json object (dictionary). The keys of the object are:

id - value is a number
firstName - value is a string
lastName - value is a string

The id is the identifier to be used when referring to the given instructor. Here is an example of an object:

```
{"id":2,"firstName":"Dr. Leland","lastName":"Beck"}
```

<http://bismarck.sdsu.edu/rateme/instructor/n>

Returns an json object. All the keys are shown in the example below. The rating key has a value that is a json object.

```
{ "id":2,"office":"GMCS  
407B","phone":"619-594-6191","email":"beck@cs.sdsu.edu","rating":{"average":5.0,"totalRa  
tings":12},"firstName":"Dr. Leland","lastName":"Beck"}
```

<http://bismarck.sdsu.edu/rateme/comments/n>

Returns the comments for the instructor with id "n". The result is a json array of objects. Each object represents one comment. The object contains keys "text" and "date". The date is the date the comment was submitted to the server. The text is the actual comment. Below is a example.

```
'[{"text":"Good Instructor","date":"10/26/11"}, {"text":"do  
it","date":"10/25/11"}, {"text":"cat","date":"10/24/11"}, {"text":"He is  
cold","date":"10/24/11"}, {"text":"He is hot","date":"10/24/11"}]'
```

Post URLs

<http://bismarck.sdsu.edu/rateme/rating/n/k>

Adds the rating "k" to the instructor with id "n". "k" is one of the values 1, 2, 3, 4, or 5.

<http://bismarck.sdsu.edu/rateme/comment/n>

Adds a comment to the instructor with id "n". The comment is the body of the post.

Grading

The assignment will be graded as follows:

Points	Item
35	Network connections
20	Table View for Instructors
20	Instructor detail view(s)
15	Posting rating & Comments
10	GUI look

What to Turn in

Create a Xcode project for the assignment. Xcode places the project in its own directory. Place the directory (and all its contents) into a zip file. We will use the course portal (<http://bismarck.sdsu.edu/CoursePortal>).