Assignment 1
This assignment cover the topic from week 1 to week 7

Purpose
To assess your ability to apply the course concepts and critical thinking skills

Due Date:
Week 9 - Saturday 31 Oct. 2015

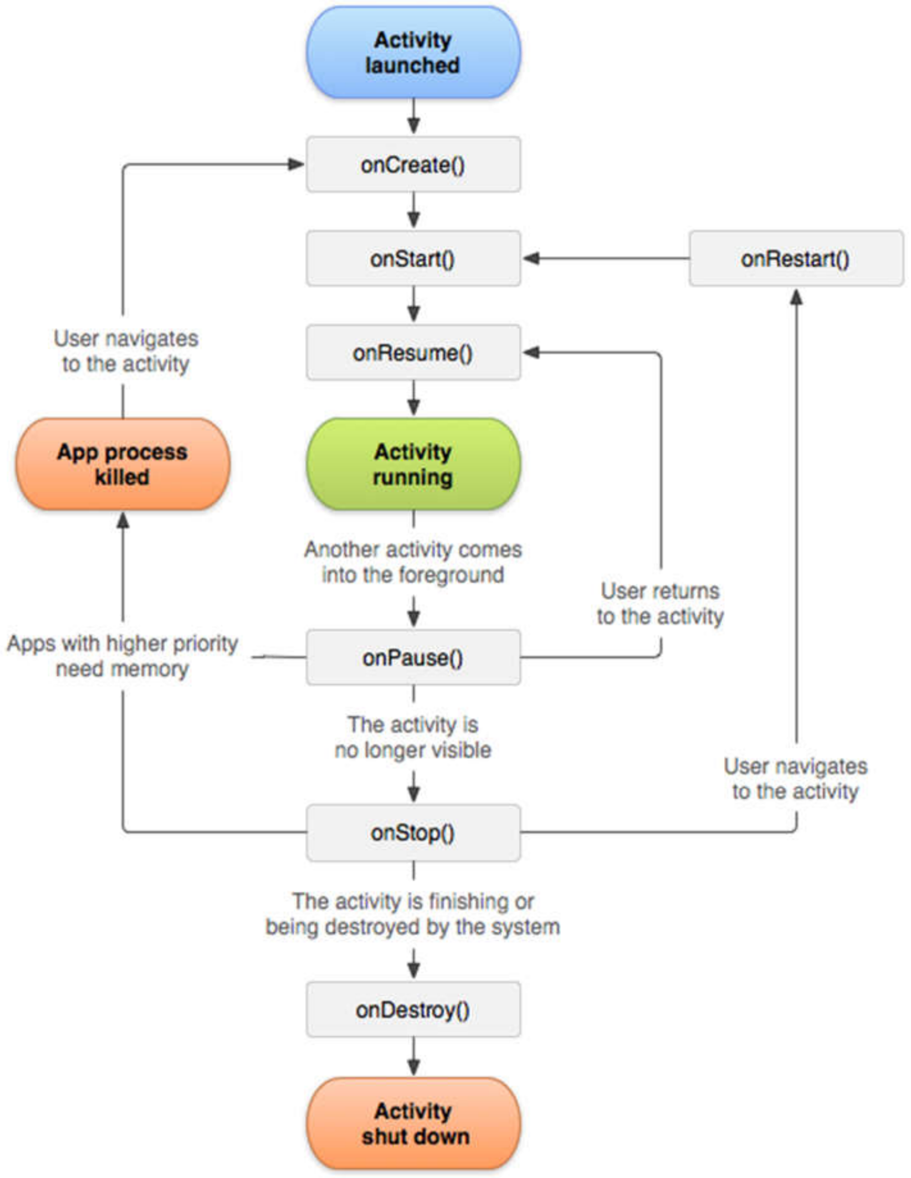
Action Items
Q1.
Explain in details the activity life cycle of an android application with the help of diagrams.

Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

- If an activity in the foreground of the screen (at the top of the stack), it is *active* or *running*.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is *paused*. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

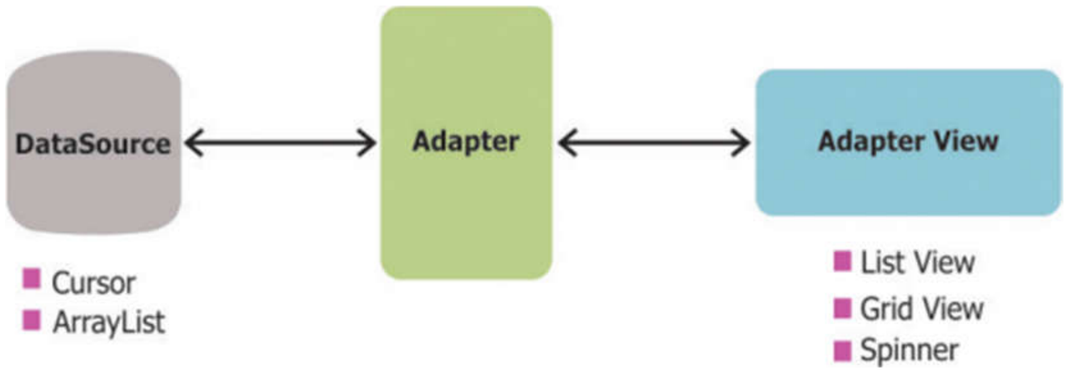
The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Q2.
What are Adapters in android, explain different adapter views

Adapters in Android are a bridge between the Adapter View (e.g. ListView) and the underlying data for that view. Lists require the use of an adapter. The adapter provides access to the data items and is responsible for creating a View for each item. A view determines how each list item is displayed. In most cases, this display is uniform for each data item. The display does not have to be uniform, but in that case, developers must implement their own adapters to create the different views.
Below is a conceptual diagram which shows the high level working of the Android Adapter:



Let us now understand the internal working of an Android Adapter and how it acts as a data pump to the adapter view. Adapters call the **getView()** method which returns a view for each item within the adapter view. The layout format and the corresponding data for an item within the adapter view is set in the **getView()** method

Q3.

Explain the Android location services and the classes required for getting GPS location.

Location and maps-based apps offer a compelling experience on mobile devices. You can build these capabilities into your app using the classes of the android.location package and the Google Maps Android API. The sections below provide an introduction to how you can add the features

Location Services

Android gives your applications access to the location services supported by the device through classes in the android.location package. The central component of the location framework is the LocationManager system service, which provides APIs to determine location and bearing of the underlying device (if available).

As with other system services, you do not instantiate a LocationManager directly. Rather, you request an instance from the system by calling getSystemService(Context.LOCATION_SERVICE). The method returns a handle to a new LocationManager instance.

Once your application has a LocationManager, your application is able to do three things:

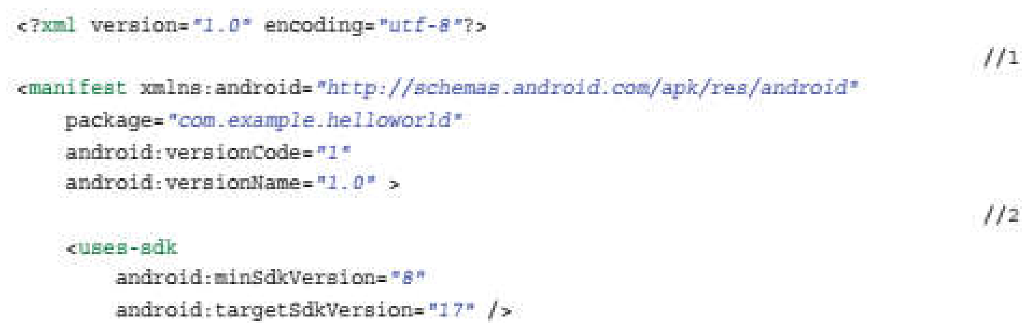
Query for the list of all LocationProviders for the last known user location.

Register/unregister for periodic updates of the user's current location from a location provider (specified either by criteria or name).

Register/unregister for a given Intent to be fired if the device comes within a given proximity (specified by radius in meters) of a given lat/long.

For more information about acquiring the user location, read the Location Strategies guide.

Q4.
The following figure shows a part of Android application manifest file.



```
<?xml version="1.0" encoding="utf-8"?>
                                                                          //1
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >
                                                                          //2
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
```

1. **Discuss in details what is meant by part number //1 and //2**
2. **In the previous figure, what changes are required to do if you made a new version of your application "1.1", also a new SDK version 18 is published.**

1.

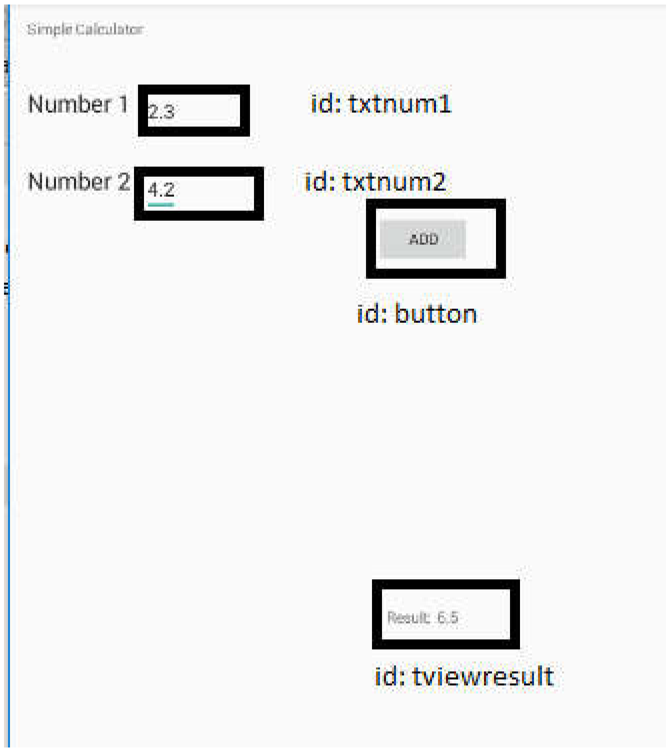
1. The <manifest> component is the root element. The attributes associated with this element define the application package, version code, and version name (as well as others).
2. The <uses-sdk> element and its attributes define the minimum and target SDKs for the app.

2.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.helloworld"
android:versionCode="2"
android:versionName="1.1" >

<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="18" />
```

Q5.
In the following activity, complete the following code in order that when a user press ADD button the sum of the two numbers in (txtnum1, txtnum2) will appear in (tviewresult).
Hint: to parse String to Float use “`Float.parseFloat`”



```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);


   Button displayButton = (Button) findViewById(R.id.button );
         displayButton.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View arg0) {
                 EditText editnum1 = (EditText) findViewById(R.id.textnum1);
                 EditText editnum2 = (EditText) findViewById(R.id.txtnum2);
                 TextView textDisplay = (TextView) findViewById(R.id.tviewresult);
                 float sum = Float.parseFloat( editnum1.getText().toString()) +
Float.parseFloat( editnum2.getText().toString())   ;
                 textDisplay.setText("Result:   " + sum);
             }
         });

}
```

Submission Instructions

Complete and submit this assignment per your professor’s instructions.

Grading Criteria

Accuracy and completion of assignment: 0 - 5 points