1. True/False and Explain:

a. True A Spinner can effectively replace any use of a group (RadioGroup) of RadioButtons.

Explanation: Both enforce a "1-of-" only selection (mutually exclusive selection)

b. False For each Activity that ends up being employed in a program, you, as a programmer, must write at least one Java class per Activity.

Explanation: We can employ activities, such as the Image Gallery, we did not personally write

c. False There are interfaces (screens) I can developing using layout XML that I cannot develop using Android-based Java directly.

Explanation: The XML and Java approaches are equivalent in which interfaces they support. The XML is actually forced into Java behind the scenes.
2. Given the following set of requirements for a mobile app desired by the Biology Department, (a) list three Views you believe are most important for ensuring the desired functionality can be implemented and (b) explain your choices.

Requirements: The Biology Department would like an app developed that provides access to a database of trees on the WFU campus that the Biology Department has recorded information about. In the Biology department database, each tree is associated with both a common name and a scientific name, and these names are stored in two different columns of the database. You are asked to write a search screen for the app which should allow a user of the app to type in a species name, indicate whether that name is the species common name or scientific name, and perform a search in the appropriate column on that name. The results returned from the search will be presented in another activity – don’t worry about that 2nd activity – just deal with designing the search activity.

(a) Three Views that are important for implementing the search screen capabilities:

- Edit Text
- RadioButton (group)
- Button

(b) Explanation of three choices in Part (a)

The "Edit Text" will support the user (typing) in a species name.

The "RadioButton" group will manage selection of common name or scientific name (one or the other).

The "Button" will trigger the actual search (under the assumption the user completed the other steps/filled in the right data in other parts of the interface).
3. The following questions ask about the notion of the Listener pattern employed by Android applications.

   a. When a listener is to be notified of an event having occurred, what is the technique for how this notification is made to the listener?

      A function (method) is called in the class doing the listening, and details of the event are passed as a parameter to that function.

   b. Explain whether or not View controls are the only source of events to be listened for in an Android app.

      No, they are not the only source. We listen for state changes to the app (onPause) and can listen for GPS/location updates.

4. I have started on an app which will allow users to learn more about degrees available at Wake Forest. Using the XML and Java code attached at the end of the test, answer the following questions:

   a. Draw in the space below what the screen will look like after the onCreate function has executed.
b. The XML employs a RelativeLayout to position the various controls. Indicate whether or not the same positioning could be achieved using a LinearLayout structure, and briefly support (provide evidence for) your answer.

Yes, a linear layout could have been employed.

The Relative Layout explicitly adds the degree spinner below the department spinner, and the button below the degree spinner.

A vertical linear layout places items below each other in the order they are listed so, department, then degree, then button.

Thus, they will look the same with either layout.

5. Assume that the Activity stack for an application is three activities high as shown below, and that the three activities are named ActivityA, ActivityB, and ActivityC:

<table>
<thead>
<tr>
<th>ActivityC</th>
<th>ActivityB</th>
<th>ActivityA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom of stack</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Given this activity stack, which Activity was the original activity launched by the app launcher, and what is your reasoning behind your answer?

Activity A was the original activity.

When new activities are started, they are placed at the top of the stack (w/ a stack, things only are added at the top). So, A must be the oldest, w/ C the newest.

b. To trigger a transition between Activities, an Intent is employed. Using an Intent usually requires several lines of code that conclude in a startActivity(intentGoesHere) statement. How does one indicate which Activity should be triggered when setting up an Intent object?

The Activity to be triggered by an Intent is specified as the 2nd parameter of the Intent constructor.

```java
Intent intent = new Intent(this, ActivityGoesHere.class);
```
c. Given the activity stack on the previous page and the activity lifecycle on the last page of the test, indicate which function listed (in the light gray boxes) in the Activity lifecycle diagram you expect to have most recently been called on ActivityB, assuming it was completely obscured by the activity it triggered. Make sure to give the reasoning behind why you chose that function.

"on Stop" should be the last method called on ActivityB. When it becomes completely obscured, it is told that it is effectively stopped. From this point, if it comes back to the foreground it can start working again.

(See the lecture 1 slides, where the different states of an Activity are discussed.)

6. When you were a kid, you probably asked your parents the question "What would happen if I dig all the way through the Earth and out the other side?". They probably answered "You'll end up in China" and then tell you to start digging to get you out of the house for a few hours. The location that would be reached if we could go straight through the Earth to the other side is called the antipodal point of the initial location. For Winston-Salem, the antipodal point is actually in the Indian Ocean off of the Australian coast!

Assume you are developing an app which acts as follows:

A user is presented a Google map in satellite mode. They are requested to center the map, by scrolling around, on the general location they are interested in finding the antipodal point of. They then hit a button which will force the app to compute the antipodal point of the point they have centered on the map. The map then animates to the antipodal point and displays that to the user.

a. For which part(s) of the process described above, if any, would access to the MapController object and its associated functions be required? In addition to listing any parts of the process, briefly explain why those parts need access to the MapController. Feel free to answer None if no parts of the process require access to the MapController, but be sure to explain that as well.

The MapController is only needed to support recentening or zooming of the map, so only the animation to the antipodal point requires the MapController. The scrolling by the user is handled automatically for us, and getting the center of the screen is done through MapView functions.
b. Given an initial point, when using standard decimal coordinates (like 36.094436, -80.240278) the latitude of the antipodal point is computed by negating the original point’s latitude. If the longitude of the original point is between 0 and -180, then the antipodal point longitude is (180 + the original longitude). Otherwise, the antipodal point longitude is (180 – the original longitude).

Given a GeoPoint variable `initialPoint`, write a small chunk of code that uses the GeoPoint API function you have been provided below to generate a new GeoPoint that correctly represents the antipodal point of `initialPoint`.

**API:**

**Constructor Summary**

<table>
<thead>
<tr>
<th>GeoPoint(int latitudeE6, int longitudeE6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructs a GeoPoint with the given latitude and longitude, measured in microdegrees (degrees * 1E6).</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(java.lang.Object object)</td>
<td></td>
</tr>
<tr>
<td>int getLatitudeE6()</td>
<td>Returns the latitude of this GeoPoint in microdegrees (degrees * 1E6).</td>
</tr>
<tr>
<td>int getLongitudeE6()</td>
<td>Returns the longitude of this GeoPoint in microdegrees (degrees * 1E6).</td>
</tr>
</tbody>
</table>

Your code:

```java
// Assume initialPoint has already been initialized
int antPodalLatitude = initialPoint.getLatitudeE6() * -1;
int currentLongitude = initialPoint.getLongitudeE6();
if (currentLongitude < 0) antPodalLongitude = (180 * 1E6) + currentLongitude;
else antPodalLongitude = (180 * 1E6) - currentLongitude;
GeoPoint targetPoint = new GeoPoint(antPodalLatitude, antPodalLongitude);
```
Activity Lifecycle (Question 5c)

- Activity starts
  - onCreate()
    - onStart()
    - onResume()
      - Activity is running
        - New activity is started
          - onFreeze()
            - onPause()
              - onStop()
                - onDestroy()

User navigates back to your activity

Other applications need memory

Your activity comes to the foreground

Your activity comes to the foreground

Your activity is no longer visible
public class Test1Problem4Activity extends Activity {

    String[] departments;
    Spinner departmentSpinner;
    ArrayAdapter<CharSequence> departmentAdapter;

    String[] degrees;
    Spinner degreeSpinner;
    ArrayAdapter<CharSequence> degreeAdapter;
    Resources resourcesInformation;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        resourcesInformation = getResources();
        departments = resourcesInformation.getStringArray(R.array.departments);
        degrees = resourcesInformation.getStringArray(R.array.degrees);

        departmentSpinner = (Spinner) findViewById(R.id.department_spiner);
        departmentAdapter = new ArrayAdapter<CharSequence>(this, android.R.layout.simple_spinner_item,departments);
        departmentAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        departmentSpinner.setAdapter(departmentAdapter);

        degreeSpinner = (Spinner) findViewById(R.id.degree_spiner);
        degreeAdapter = new ArrayAdapter<CharSequence>(this, android.R.layout.simple_spinner_item,degrees);
        degreeAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        degreeSpinner.setAdapter(degreeAdapter);
    }
}