Mini Test

1. (Sensor Framework)
A)

a) List available sensors on a device

```
import android.hardware.SensorManager;

SensorManager sensorMgr = (SensorManager) getSystemService(SENSOR_SERVICE);
List<Sensor> sensors;
sensors = sensorMgr.getSensorList(Sensor.TYPE_ALL);     // get sensor list

        String[] sensor_names = new String[sensors.size()];
        int i = 0;                                              // get sensor names
        for (Sensor sensor : sensors) {
        sensor_names[i] = sensor.getName();
        i+=1;
        }
// the available sensors are now listed in the sensor_names string-array
```

b) Retrieve the value range of a specific sensor

```
//assuming the variable sensor is of type Sensor and is assigned to a sensor
        float max_value = sensor.getMaximumRange();
float min_value = -max_value;
```

c) Register for monitoring accelerometer sensor changes at the maximum available rate

```
import android.hardware.SensorManager;

SensorManager sensorMgr = (SensorManager) getSystemService(SENSOR_SERVICE);
Sensor sensor = sensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorMgr.registerListener(this, sensor, SensorManager.SENSOR_DELAY_FASTEST);
```

B)

In the android developers guide, there's an important note on the SensorEventListener page, stating:
NOTE: The application doesn't own the event object passed as a parameter and therefore cannot hold on to it.
The object may be part of an internal pool and may be reused by the framework.

The problem is that the onSensorChanged() event can be invoked concurrently because it is an event listener.
Therefore the event object can change during the execution of the onSensorChanged event and it is therefore
possible that both cases in the switch statement are true: First, the onSensorChanged event is called from the
ACCELEROMETER. While the values are being logged, the onSensorChanged event of PROXIMITY is called and
therefore the event.values change and are logged wrong in the accelerometer onSensorChanged call.
The event object should always be copied before performing operations with it.

2. (Activity lifecycle)

The activity must be in state Resumed, Paused or Stopped. When you transition to these states `onResume()`,
`onPause()` and `onStop()` are called respecively.

3. (Resources)

Strings should be defined in the /res/values/strings.xml file. The advantage of this is that all strings
are declared at one central location, if something needs to be changed the user knows where. No need to
search through all the code. And if you want to translate your app into another language you only have to
provied an additional file containing all the strings translated to the additional language.

4. What are Intents? What are they used for? What is the difference between Explicit
Intents and Implicit Intents?

An Intent is a messaging object which can be used to request an action from another app component.
The three fundamental use-cases are starting an activity, starting a service and delivering a broadcast.
Explicit intents specify the component to start by (fully-qualified class) name.
Typically used to start a component in your own app.
Implicit intents do not name a specific component, but instead declare a general action to perform
(e.g. show the user a location on a map), which allows a component from another app to handle it.

5. (Service lifecycle) State whether each of the following sentences is true or false:

a) false: A service can stop itself, be stopped by another component or by the system to free resources before all the work is done.
b) false: an unbound service runs for itself until the job is done or the service stopped. Only bound services can interact with multiple client processes.
c) true: if no component is bound to the service it gets destroyed.
d) false: services are started in the main thread by default

6. (AndroidManifest file) List the needed tags.

1. `<service android:enabled="true" android:name=".LocationService" />`
   otherwise the service is not usable
2. `<uses-permission android:name="android.permission.SEND_SMS"/>`
   sending sms needs a special permission
3. `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>`
   getting the exact location needs a special permission (ACCESS_COARSE_LOCATION for approximated location)