Android Basics 1.1 K

Name: _____

1. What language is used to code Android GUIs? . . . . . .
2. What are four different types of views?

3. Write 10 attributes of a TextView that can be changed:

4. Fill in the blanks to make the TextViews shown.

(a) The Title

Font is sans-serif-light, size is 30sp
Color is holo_red_dark
Padding is 20dp

```
<TextView
    android:text="_____"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fontFamily="_____"
    android:textColor="@android:color/_____"
    android:textSize="_____"
    android:padding="_____"
    />
```

(b) The Subtitle

Font is sans-serif-light, size is 35sp
Color is holo_blue_dark
Padding is 25dp

```
<_____
    android:text="_____"
    _____:layout_width="match_parent"
    android:layout_____="wrap_content"
    _____:fontFamily="_____"
    android:textColor="@android:color/_____"
    _____:textSize="_____"
    android:padding="_____"
    />
```

5. Fill in the blanks to make the EditTexts shown.

(b) Phone Number

The hint is shown above.
Font is sans-serif-light, size is 40sp

```
<_____
    android:hint="_____"
    android:layout_____="match_parent"
    _____:layout_height="wrap_content"
    _____:fontFamily="_____"
    android:textSize="_____"
    />
```

(b) Date

The hint is shown above.
Font is sans-serif-light, size is 20sp

[you write the code yourself for the Date EditText]

6. Fill in the blanks to make the buttons shown.

(a)

font size: 30dp
padding of 20dp
white on holo_purple

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="_____"
    android:textSize="_____"
    android:padding="_____"
    android:background="@android:color/_____"
    android:textColor="@android:color/_____"
    />
```

(b)

font size of 30dp
padding of 20dp
holo_red on black

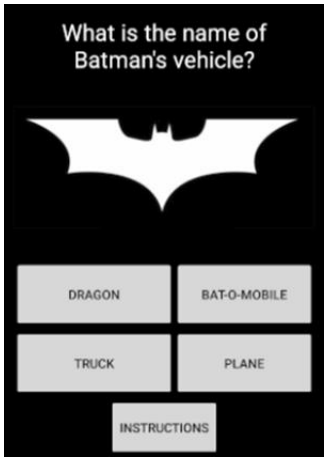
```
<_____
    android:text="_____"
    android:layout_____="wrap_content"
    _____:layout_height="wrap_content"
    android:textSize="_____"
    android:padding="_____"
    android:background="@android:color/_____"
    android:_____="@android:color/_____"
    />
```

7. Fill in the word or term that is described.

	(a) A view that the user can type in.
	(b) A view that the user can click on.
	(c) A view that holds a picture.
	(d) A view that holds a logo.
	(e) A view that holds an instruction.
	(f) A width that makes the view the same size as the parent.
	(g) A width that makes the view the same size as its text.

8. Identify the number of each type of widget shown below.

_____ Button
_____ ImageView
_____ TextView
_____ EditText



_____ Button
_____ ImageView
_____ TextView
_____ EditText

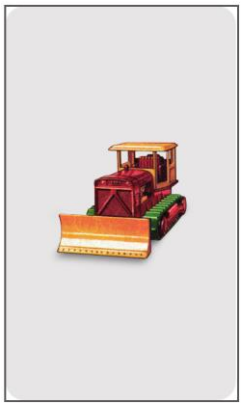
_____ Button
_____ ImageView
_____ TextView
_____ EditText

Android View Sizes

1.2 C

Name: _____

1. Select **match_parent** or **wrap_content** for each ImageView's layout_width and layout_height.



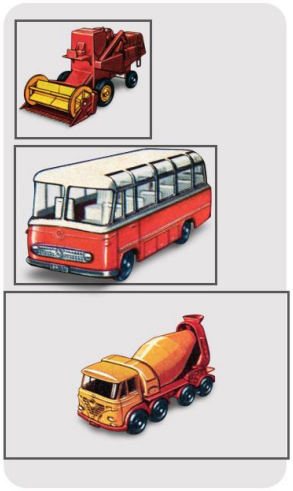
```
<ImageView
android:layout_width=
"_____"
android:layout_height=
"_____"
android:src=
"@drawable/bulldozer"
/>
```

```
<ImageView
android:layout_width=
"_____"
android:layout_height=
"_____"
android:src=
"@drawable/crane"
/>
```

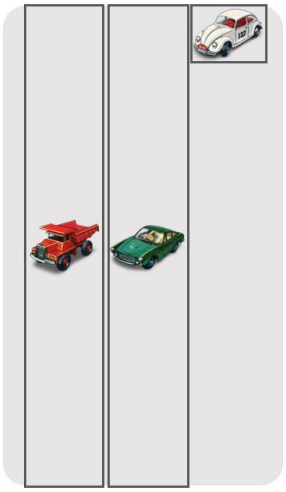
```
<ImageView
android:layout_width=
"_____"
android:layout_height=
"_____"
android:src=
"@drawable/car"
/>
```

```
<ImageView
android:layout_width=
"_____"
android:layout_height=
"_____"
android:src=
"@drawable/dumptruck"
/>
```

2. Select **match_parent** or **wrap_content** for each ImageView's layout_width and layout_height.
The ImageViews should be filled in using the order they appear on the screen top/down or left/right.



```
layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"
```



```
layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"
```

```
layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"

layout_width="_____"
layout_height="_____"
```

3. Consider the android screen shown. Enter match_parent, wrap_content or dimensions for each view.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_gravity="center_horizontal"
  android:orientation="vertical">

  <_____________________
    android:layout_width="_____________________"
    android:layout_height="_____________________"
    android:text="Tractors"
    android:textAlignment="center"
    android:padding="40dp"
    android:textSize="40sp"/>

  <_____________________
    android:layout_width="_____________________"
    android:layout_height="_____________________"
    android:src="@drawable/fordtractor"/>

  <_____________________
    android:layout_width="_____________________"
    android:layout_height="_____________________"
    android:layout_gravity="center_horizontal"
    android:text="Instructions"
    android:textSize="20dp"/>

  <_____________________
    android:layout_width="_____________________"
    android:layout_height="_____________________"
    android:layout_gravity="center_horizontal"
    android:text="Enter"
    android:textSize="20dp"/>

</LinearLayout>
```

4. Circle and correct one problem in each View.

(a)
```
<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Motorcycle"
  android:padding="25dp"
  android:textSize="20sp"
```

(b)
```
<JButton
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Continue"
  android:textSize="40sp"/>
```

(c)
```
<TextView
  android:layout_width="400dp"
  android:layout_height="100dp"
  android:text="Driving"
  android:textSize="20sp"/>;
```

(d)
```
<Image
  android:layout_height="wrap_content"
  android:layout_width="match_parent"
  android:src="@drawable/snowblower"/>
```

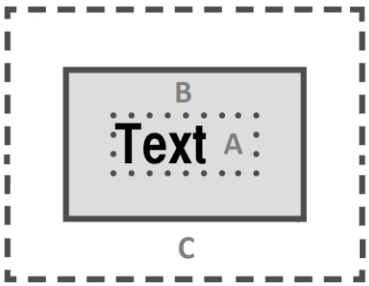
(e)
```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="First Name"
    android:layout_height="wrap_content"
    android:layout_gravity="center"/>
```

(f)
```
<Button
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:text="Introduction"
  android:textSize="24sp"/>
```

Margins, Padding and Spacing 1.3

Name: _____

1. What does DP stand for? .....
2. What does SP stand for? .....
3. Match each letter with the XML code that modifies it.



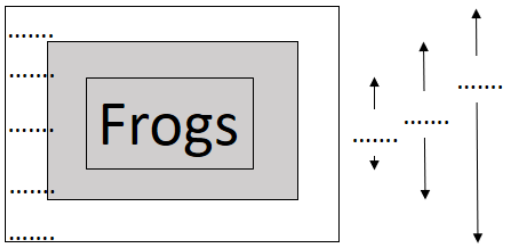
..... `android:layout_padding="10dp"`

..... `android:layout_margin="10dp"`

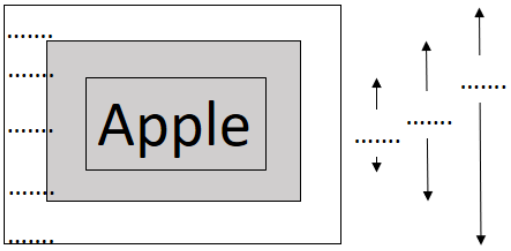
..... `android:textSize="20sp"`

4. Label the 2 height dimensions based on the textview's code.

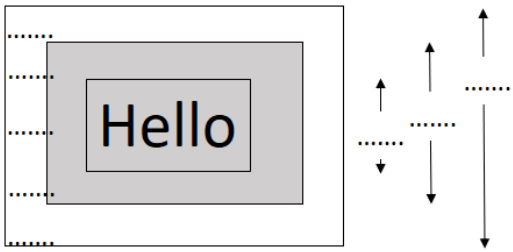
(a)
```
<TextView
    android:text="Frog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="40sp"
    android:padding="50dp"/>
```



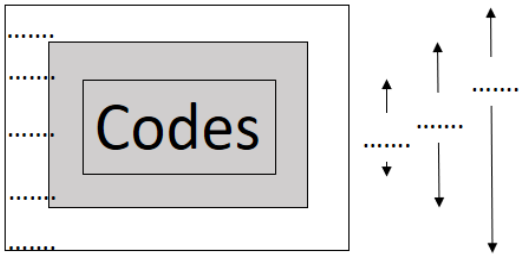
(b)
```
<TextView
    android:text="Apple"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:padding="10dp"
    android:layout_margin="40dp"/>
```



(c)
```
<TextView
    android:text="Hello"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="60sp"
    android:padding="20dp"
    android:layout_margin="10dp"/>
```




```
<TextView
    android:text="Codes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="100sp"
    android:layout_margin="25dp"/>
```



5. Circle and correct 5 errors in this button's code.

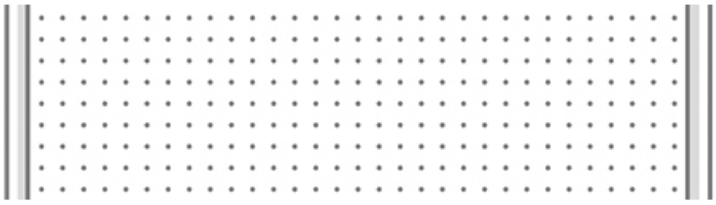
30dp text size
5dp padding

```
<JButton
    android:layout_height="match_parent"
    android:layout_height="wrap_content"
    android:text="Banana"
    android:textSize="30sp"
    android:padding="5dp";
>
```

6. Draw these widgets using colour. The width of the screen is shown. Each dot square is 10 dp.

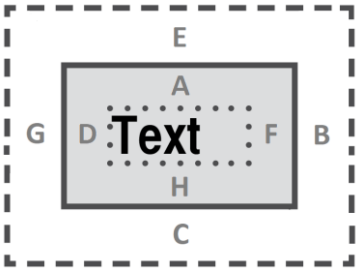
(a)
```
<TextView
   android:text="Mouse"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#00FF00"
   android:textSize="50sp"
   android:padding="10dp"/>
```

(b)
```
<TextView
   android:text="Banana"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#FF00FF"
   android:textSize="30sp"
   android:padding="20dp"/>
```



(c)
```
<TextView
   android:text="Greetings"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="#00FFFF"
   android:textSize="40sp"
   android:padding="20dp"/>
```

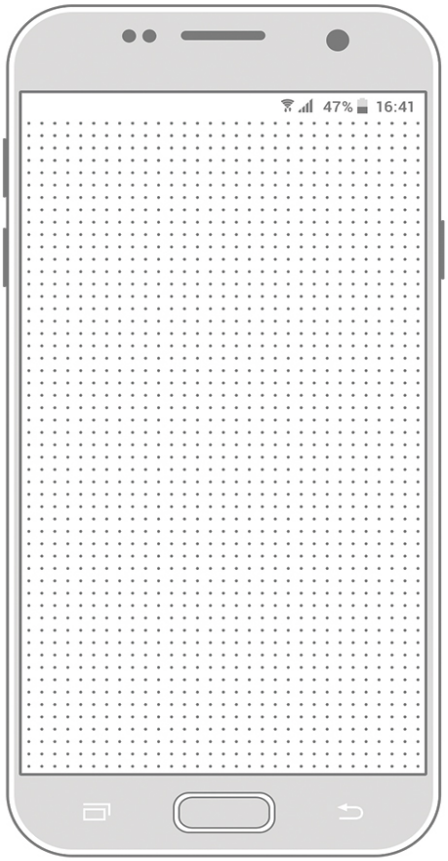
7. Match each letter with the XML code that modifies it.



..... android:layout_marginTop="10dp"
..... android:layout_marginRight="10dp"
..... android:layout_marginBottom="10dp"
..... android:layout_marginLeft="10dp"
..... android:layout_paddingTop="10dp"
..... android:layout_paddingRight="10dp"
..... android:layout_paddingBottom="10dp"
..... android:layout_paddingLeft="10dp"

8. Draw what is produced on the screen by this code. Use colour. Assume a dot square is 10 dp.

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"  >
    <TextView
        android:text="Algorithms"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00FFFF"
        android:textSize="40sp"
        android:padding="50dp"/>
    <TextView
        android:text="QuickSort"
        android:layout_width="140dp"
        android:layout_height="100dp"
        android:background="#FF0000"
        android:textSize="40sp" />
    <TextView
        android:text="Bubblesort"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#FFFF00"
        android:textSize="40sp"
        android:padding="10dp"/>
    <TextView
        android:text="Mergesort"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00FF00"
        android:textSize="40sp" />
</LinearLayout>
```



Linear Layouts

1.4 C

Name: _____

1. Identify two ways the XML syntax for layouts is different from XML for a view?

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</LinearLayout>
```

```
<TextView
    android:text="The Title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="50dp"
/>
```

a) .....

b) .....

2. Identify two ways that the XML syntax for layouts is the same as the XML for a view.

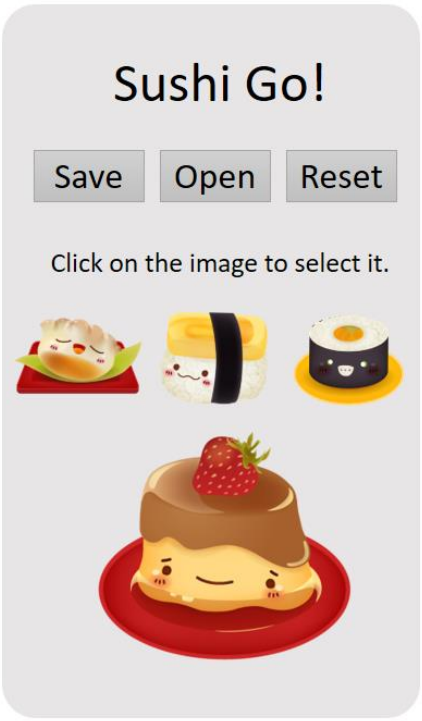
a) .....

b) .....

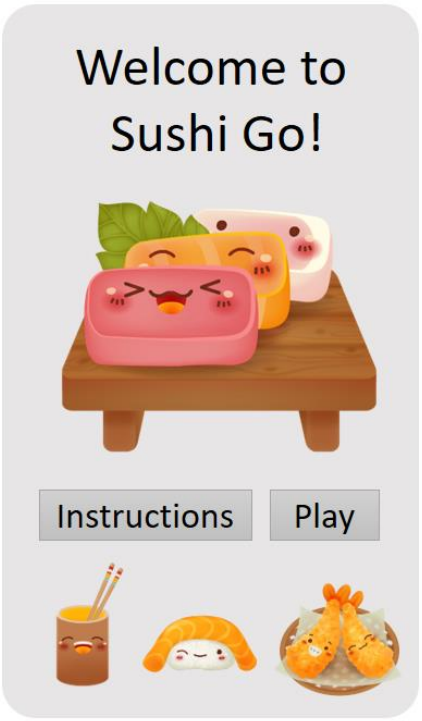
3. What are two linear layout orientations? ..... and .....

4. Put a rectangle around the outer and internal linear layouts in each screen.

3 linear layouts



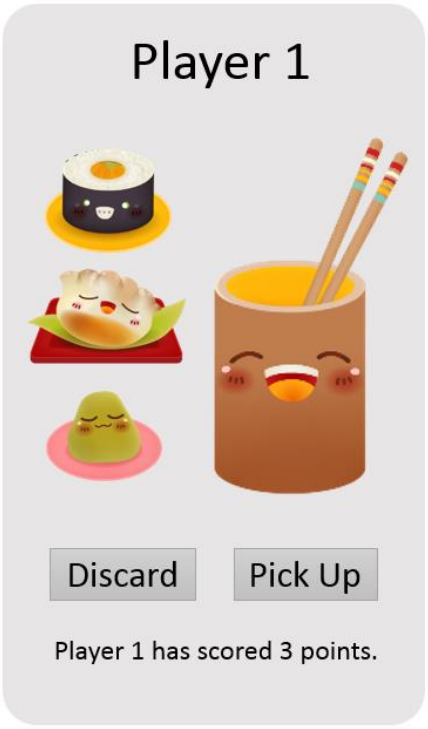
3 linear layouts



2 linear layouts

5. Put a rectangle around the outer and internal linear layouts in each screen.

4 linear layouts



4 linear layouts

3 linear layouts

6. Look at the picture of the app. Add in the Linear Layout to format the (greatly reduced) code as the picture.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#049591"
    android:orientation="vertical">

    <ImageView
        android:src="@drawable/back" />

    <TextView
        android:text="Clue: Something that makes people sneeze."/>




    <EditText
        android:hint="Answer?"/>

    <Button
        android:text="check"/>


    <TextView
        android:text="Answer the question. Check to verify."/>

    <TextView
        android:text="You have solved 0 puzzles in 0 tries."/>

</LinearLayout>
```

Android Buttons that Work 1.5 T

Name: _____

1. Circle the environment described.

(a) The screen is designed.	XML	Java	Emulator
(b) This loads the entire Android OS.	XML	Java	Emulator
(c) The events are coded.	XML	Java	Emulator
(d) This file should never be entirely deleted.	XML	Java	Emulator
(e) The attributes are selected.	XML	Java	Emulator
(f) The ifs, loops, and methods are written.	XML	Java	Emulator
(g) This is where your code actually runs.	XML	Java	Emulator
(h) This file can be entirely deleted.	XML	Java	Emulator
(i) This is where you can click on buttons.	XML	Java	Emulator

2. Put these run-time events in order:
   _____ While the app is running, the user clicks the button.
   _____ The code in the sushi method updates the screen accordingly.
   _____ The app opens. The button is created on the screen. It's onClick is sushi.
   _____ The app's GUI sees the button was pressed, and calls the sushi method.

3. Fill in the blanks to make each of the following buttons in XML and their associated method in MainActivity:

(a) 

Method name: ribbit,
Text Size: 30 sp
Padding: 20dp

```
XML:
<Button
    android:id="@+id/frog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="_____"
    android:padding="_____"
    android:text="_____"
    android:textSize="_____" />

MainActivity:
   public void _____(View view){
   }
```

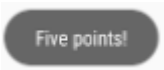
(b) 

Method name: meow
White on Black
Text Size: 50 sp

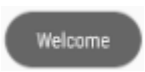
```
XML:
<Button
    android:id="@+id/cat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="_____"
    android:text="_____"
    android:background="_____"
    android:textColor="_____"
    android:textSize="_____" />

MainActivity:
   public void _____(View view){
   }
```

4. Toasts are messages that appear on the screen briefly (like showStatus). Fill in the code to create these toasts.


```
Toast.makeText(getApplicationContext(), "_____", Toast.LENGTH_SHORT).show();
```




```
Toast.makeText(getApplicationContext(), "_____", Toast.LENGTH_SHORT).show();
```




```
Toast.makeText(getApplicationContext(), "_____", Toast.LENGTH_SHORT).show();
```

5. ImageViews can also be used as buttons. Fill in the XML to make the fishes clickable:

(a) 

Picture: blue.png
Method: siesta

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="_________________"
    android:layout_margin="20dp"
    android:onClick="_________________"
    android:src="@drawable/_________________" />
```

(b) 

Picture: green.png
Method: swim

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="_________________"
    android:layout_margin="20dp"
    android:onClick="_________________"
    android:src="@drawable/_________________" />
```

(c) 

Picture: yellow.jpg
Method: sun

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="_________________"
    android:layout_margin="20dp"
    android:onClick="_________________"
    android:src="@drawable/_________________" />
```

(d) 

Picture: red.jpg
Method: kelp

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="_________________"
    android:layout_margin="20dp"
    android:onClick="_________________"
    android:src="@drawable/_________________" />
```

6. Using the above image views, fill in the Java file.

```
package ca.gorskicompsci.www.swimmers;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    __________ __________ ___________(_________ _________){
        Toast.makeText(getApplicationContext(),
                "Blue fishes enjoy a siesta.",
                Toast.LENGTH_SHORT).show();
    }

    public void swim(View view){
        Toast.makeText(getApplicationContext(),
                "_____________________________",
                Toast.LENGTH_SHORT).show();
    }

    __________ __________ ___________(_________ _________){
        Toast.makeText(getApplicationContext(),
                "Yellow fishes love warm water.",
                Toast.LENGTH_SHORT).show();
    }

    __________ __________ ___________(_________ _________){
        Toast.makeText(getApplicationContext(),
                "Red fishes eat kelp.",
                Toast.LENGTH_SHORT).show();
    }
}
```

Inflation to Java Code

1.6 K

Name: _____

1. Write the code you would need in the java file to change the TextView as instructed.

You have 5 points **android:id="@+id/points"** change to: You have 10 points.	`TextView _____ = (TextView) findViewById(R.id._____);` `_____.setText("_____");`
Click the fish to start **android:id="@+id/update"** change to: You Win!.	`TextView _____ = (TextView) findViewById(R.id._____);` `_____.setText("_____");`
It costs $0.00 **android:id="@+id/cost"** change to: It costs $4.56.	`TextView _____ = (TextView) findViewById(R.id._____);` `_____.setText("_____");`

2. For each set of code, start at the first picture. When it is clicked, change it to the second.

stego.png

brono.png

```
In XML:
<ImageView
    android:id="@+id/dino"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:onClick="dinoClick"
    android:src="@drawable/_____" />

In MainActivity.java:
public void _____(View view) {
    ImageView _____ = (ImageView) findViewById(R.id._____);
    _____.setImageResource(R.drawable._____);
}
```

nopants.png

pants.png

```
In XML:
<ImageView
    android:id="@+id/crazy"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:onClick="dressUp"
    android:src="@drawable/_____" />

In MainActivity.java:
public void _____(_____ _____) {
    ImageView _____ = (_____) findViewById(R.id._____);
    _____.setImageResource(R.drawable._____);
}
```

happy.png

mad.png

```
In XML:
<ImageView
    android:id="@+id/emotions"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:onClick="change"
    android:src="@drawable/_____" />

In MainActivity.java:
public void _____(_____ _____) {
    _____ _____ = (ImageView) findViewById(R.id._____);
    _____.setImageResource(R.drawable._____);
}
```

3. For each EditText, get the data out of it and then clear it.

Name
`android:id="@+id/name"`
	`EditText _____ = (EditText) findViewById(R.id._____);`
	`String user = _____.getText().toString();`
	`_____.setText("");`

Age
`android:id="@+id/age"`
	`EditText _____ = (_____) findViewById(R.id._____);`
	`int userage = Integer.parseInt(_____.getText().toString());`
	`_____.setText("");`

amount
`android:id="@+id/amt"`
	`_____ _____ = (EditText) _____(R.id._____);`
	`int quan = Integer.parseInt(_____._____().toString());`
	`_____._____("");`

4. Fill in the blanks on the following app so it switches the fish picture when clicked.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/swimmer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="switchMe"
        android:src="@drawable/blue"
        android:padding="100dp"/>

</LinearLayout>
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void _____(_____ _____){
        int n = (int)(Math.random()*4);
        ImageView fish = (_____) findViewById(R.id._____);
        if(n==1)
            _____.setImageResource(R.drawable._____);
        else if(n==2)
            _____._____(R.drawable.red);
        else if(n==3)
            _____.setImageResource(R._____.green);
        else
            _____.setImageResource(_____.drawable.yellow);
    }
}
```

5. In Android, why do we need to use findViewById? (2 points)

.....

.....

.....

Wind Speed Calculations (Buttons) 1.7 K

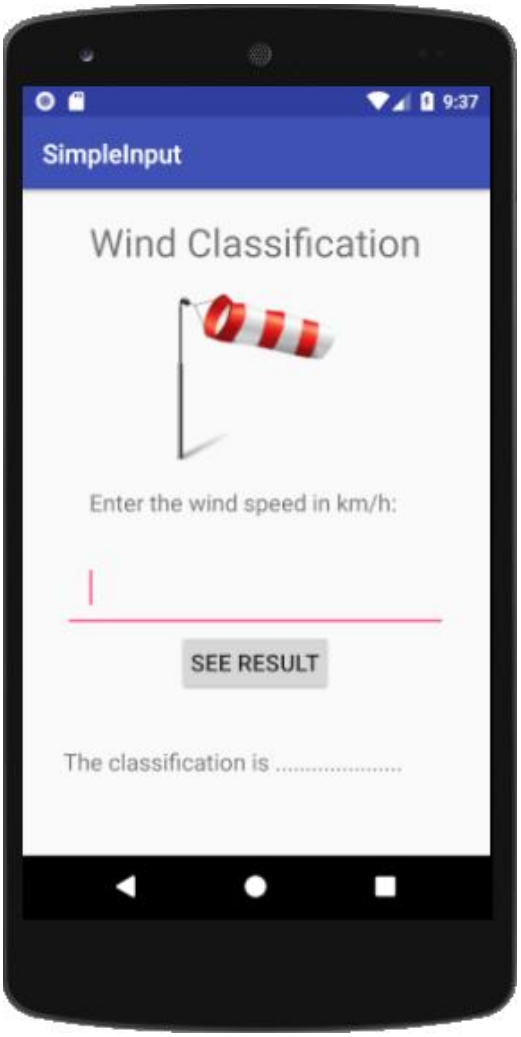
Name: _____

1. Answer the following questions about the app shown.

(a)	Name of the App		(d)	Number of Views	
(b)	Number of EditTexts		(e)	Number of ImageViews	
(c)	Number of Buttons		(f)	Number of TextViews	

2. Fill in the blanks to create the app shown.

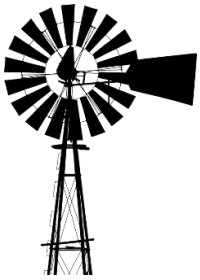
```
<?xml version="1.0" encoding="utf-8"?>
<_____ xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:orientation="vertical">

    <TextView
        android:layout_width="_____"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:text="_____"
        android:textSize="30dp" />

    <_____
        android:layout_width="wrap_content"
        android:_____="wrap_content"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/wind" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:text="_____"
        android:textSize="18dp" />

    <_____
        android:id="@+id/input"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:_____="20dp"
        android:textSize="20dp"
        android:layout_gravity="center_horizontal"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:_____="result"
        android:padding="10dp"
        android:text="_____"
        android:textSize="18dp" />

    <TextView
        android:_____="@+id/output"
        android:_____="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="40dp"
        android:_____="The classification is ....."
        android:textSize="18dp" />
</_____>
```



3. Fill in the code to make it work.

```
package ca.gorskicompsci.www.simpleinput;
import android.support.v7.app.AppCompatActivity; import android.os.Bundle;
import android.view.View; import android.widget.EditText; import android.widget.TextView;

 public class MainActivity extends AppCompatActivity {
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
     }

     public void result(View view) {
         TextView output = (TextView) findViewById(R.id.output);
         EditText input = (EditText) findViewById(R.id.input);
         int num = Integer.parseInt(input.getText().toString());
         if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else
             output.setText("_____________________________________");
     }
 }
```

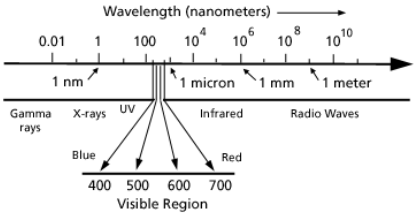
Wind Scale	Beaufort
< 5	Calm
< 11	Light Wind
< 19	Gentle Wind
< 28	Moderate
< 49	Strong
< 74	Gale
< 117	Storm
117+	Hurricane



4. Re-code the app to match the output table shown.

```
package ca.gorskicompsci.www.simpleinput;
import android.support.v7.app.AppCompatActivity; import android.os.Bundle;
import android.view.View; import android.widget.EditText; import android.widget.TextView;

 public class MainActivity extends AppCompatActivity {
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
     }

     public void result(View view) {
         TextView ______ = (TextView) findViewById(R.id.output);
         EditText input = (__________) findViewById(R.id.input);
         ____ num = Integer.parseInt(input.getText().toString());
         if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else if (num < _______)
             output.setText("_____________________________________");
         else
             output.setText("_____________________________________");
     }
 }
```

Wavelength	Colour
< 400	Ultraviolet
< 450	Violet
< 500	Blue
< 570	Green
< 590	Yellow
< 610	Orange
< 700	Red
700+	Infrared



Electromagnetic Spectrum

Testing

1.8 T

Name: _____

1. You have been asked for a street name. Classify each of the following as either a white or black box test case.

Black White a) These test cases are chosen by looking at the code.

Black White b) These test cases are selected without looking at the code.

Black White c) Average cases: In Brampton, there is a street named Hurontario

Black White d) Short cases: In rural Ontario, there are streets named RR1

Black White e) The loop to get addresses is run exactly once.

Black White f) Avoid the loop to get addresses entirely.

Black White g) Long cases: a German street is named Bischöflich-Geistlicher-Rat-Josef-Zinnbauer-Straße

Black White h) Each branch of an if about addresses is run.

Black White i) Boundary cases: In England, there is a street named Street

2. Why are there so many kinds of testing? (white, black, alpha, beta, regression…)

. . . . . .

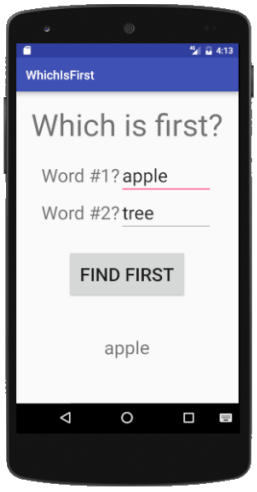
. . . . . .

. . . . . .

. . . . . .

3. Black box test each set of code.

(a)




```
//Pre: two words are entered
//Post: the word that is alphabetically first is
        displayed under the button.
```

Short Data	
Long Data	
Average Data	
Boundary Case	

(b)

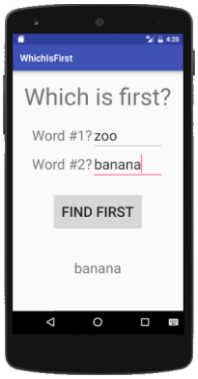

```
//Pre: the sequence position is entered,
       0 is the first position
//Post: the Fibonacci number in that position of the
       sequence is displayed.
//Sequence is: 1  1  2  3  5  8  13  21  34  55
```

Short Data	
Long Data	
Average Data	
Boundary Case	

4. White box test each set of code.

(a)

```
public void findFirst(View view){
    EditText word1 = (EditText) findViewById(R.id.word1);
    EditText word2 = (EditText) findViewById(R.id.word2);
    TextView first = (TextView) findViewById(R.id.first);
    String w1 = word1.getText().toString();
    String w2 = word2.getText().toString();
    if(w1.compareTo(w2)<0)
        first.setText(w1);
    else if(w2.compareTo(w1)<0)
        first.setText(w2);
    else
        first.setText("Equal");
}
```



Test Case	Example Data (a pair of words)

(b)

```
public void fib (View view){
    TextView result = (TextView) findViewById(R.id.result);
    EditText pos = (EditText) findViewById(R.id.pos);
    int end = Integer.parseInt(pos.getText().toString());
    int a = 1;
    int b = 1;
    for(int i=0; i < end; i++){
        int temp = a+b;
        a=b;
        b=temp;
    }
    if(end<0)
        result.setText("Only positive");
    else if(end<2)
        result.setText("1");
    else
        result.setText(a+"");
}
```

Test Case	Example Data (a sequence number)

5. Why is testing important? Use a specific example.

.....

.....

.....

.....

PARC Principles of Design 1.9 C

Name: _____

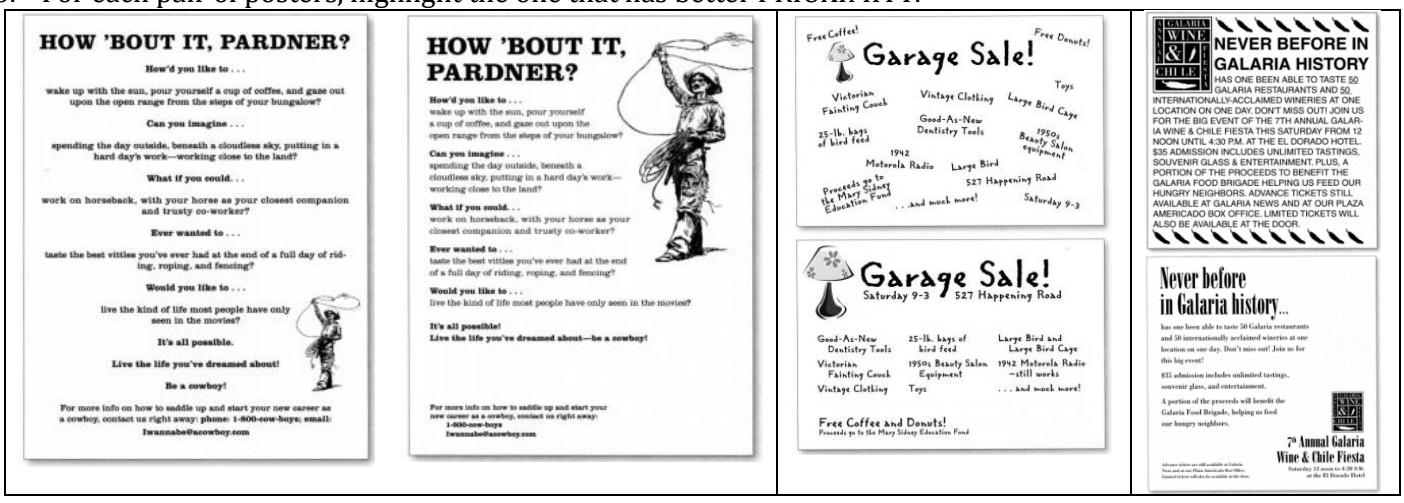
1. Match the PARC principle with description. (Proximity, Alignment, Repetition, Contrast)

	a) The element that is the most important, or the starting point, stands out because it is formatted differently than all of the other elements in the app.
	b) Visual elements (fonts, colours, shapes, picture styles, padding sizes) should repeat throughout the app.
	c) Nothing is placed on the screen arbitrarily. Every element should line up with other elements on the screen.
	d) Items relating to each other should be grouped together. These groups should be visually separated from other groups.

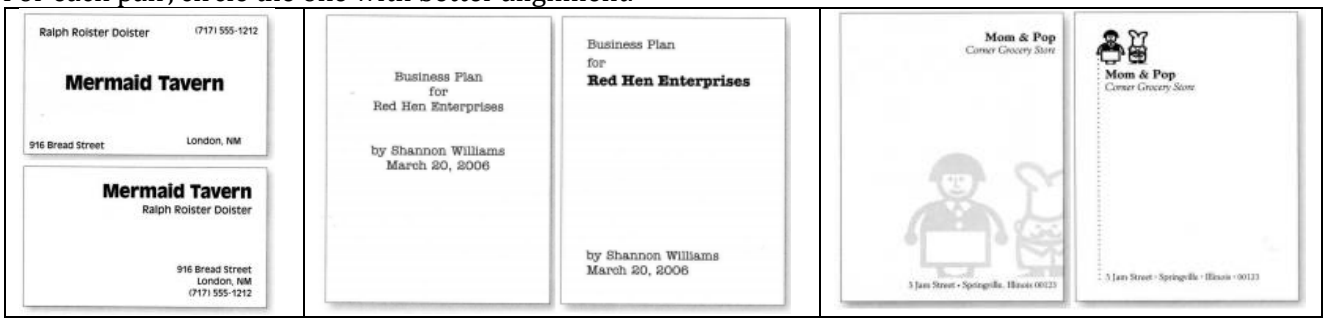
2. Match each description to a PARC principle. (Proximity, Alignment, Repetition, Contrast)

	a) All buttons on the screen are light blue with a textSize of 30sp, margins of 10dp and padding of 10dp.
	b) Everything on the screen is centered.
	c) The title is larger than the other elements on the screen. It is also a different colour and is bolded so that it stands out.
	d) The score and status information is separated from the rest of the screen using a linear layout and padding.
	e) The screen has black writing on white buttons, except for the start button. That button is white writing on the black background so that it stands out.
	f) The 5 buttons at the bottom of the screen are in a linear layout. They are separated from the other elements of the screen using margins.
	g) Everything on the screen is right aligned.
	h) The background image fades into the background and isn't busy. This is so that the items on top of it (which are needed for the game) can be easily seen.

3. For each pair of posters, highlight the one that has better PRIOXIMITY.



4. For each pair, circle the one with better alignment.



5. Which elements are repeated in this tea party invitation?

a) .....

b) .....

c) .....

d) .....

6. Which element is contrasted in this tea party invitation?

a) .....

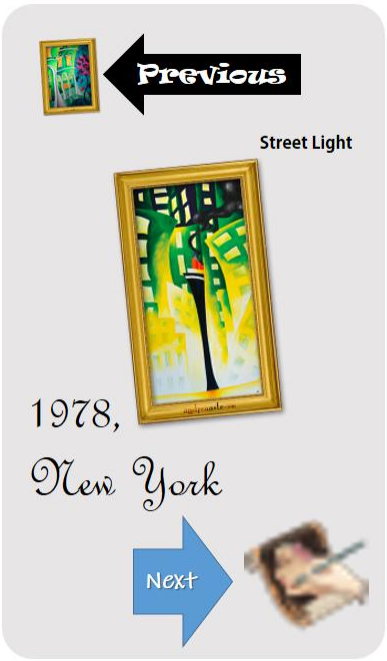
7. Circle the icons that can be used in the same app - because they **repeat** the same style.

8. Which describes a colour scheme? (Circle the best answer)
   (a) Black background with bright colours in widgets.
   (b) Titles in #DE45FF, Buttons in #8E2E44, Writing in #DE6040; Contrasting Background in #34EE20,
   (c) Colourful. You know, the rainbow.
   (d) All different shades of green.

9. Match the PARC principle to the reason it is important. (Proximity, Alignment, Repetition, Contrast)

	a) Careful placement of elements on the screen creates a clean, sophisticated look. It unifies the page.
	b) Visual units separated by space help organize information, reduce clutter and give the page clear structure.
	c) One very different element will draw the user's eye to that element on the screen. This provides a useful starting point for the user.
	d) Duplication and consistency makes the app have a cohesive look and feel. It is a conscious effort to unify all parts of a design.

10. For each PARC principle, identify an error in the screen. Explain it briefly.



Proximity
Alignment
Repetition
Contrast