Layouts

AIM: Create an Android application to design screens using different layouts and UI including Button, Edittext, Textview, Radio Button, CheckBox etc.

Theory:

An Android layout is a class that handles arranging the way its children appear on the screen. All of the layouts inherit from **ViewGroup** (which inherits from **View**) so you can nest layouts. You could also create your own custom layout by making a class that inherits from **ViewGroup**.

There are various in android:

1. Linear Layout :

LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

Following are the important attributes specific to LinearLayout −

- android:id

  This is the ID which uniquely identifies the layout.

- android:divider

  This is drawable to use as a vertical divider between buttons. You use a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

- android:gravity

  This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.

- android:orientation

  This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal

2. Table Layout:

Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

TableLayout containers do not display border lines for their rows, columns, or cells.

- android:collapseColumns

  This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.

- android:shrinkColumns

  The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.

- android:stretchColumns

  The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5.

3. Constraint Layout

Constraint Layout is a ViewGroup (i.e. a view that holds other views) which allows you to create large and complex layouts with a flat view hierarchy, and also allows you to position and size widgets in a very flexible way. It was created to help reduce the nesting of views and also improve the performance of layout files.

Terminology used:

Constraints: are the basic building blocks of constraint layout.

Anchor point or Contsrint Handle: we can see circles in each side of text view.

Start/left, right/end, top and bottom

Anchor point is to create constraint/relationship between widgets.

Baseline anchorpoint : used to constraint baseline of the text of the widget.(ie to align text).Capsule represent baseline.

Questions:

1. Create android application to demonstrate Linear Layout.
2. Create android application to demonstrate Table Layout.
3. Create android application to demonstrate Constraint Layout.

NOTE:

(Attach print out of XML file and Output)

Theory and Java code must be handwritten.