SUBJECT: ANDROID DEVELOPMENT **NAME:**	
CLASS:	**ROLL NO:**
SEMESTER/YEAR:-	**EXAM NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED:-	**REMARKS:**

EXPERIMENT NO – 5

TITLE: Design a mobile application to create home page using grid layout

OBJECTIVE:

1. To study and design a mobile application to create home page using grid layout

PREREQUISITE:

Students should be aware of Android Studio platform

TOOLS USED:

1. Android studio (Electric Eel)

THEORY:

Grid Layout

Android provides you with a feature to style your app screens using various types of layouts. Some of the most commonly used layouts are LinearLayout, Constraint Layout, RelativeLayout, and GridLayout.

Suppose you need to display the elements linearly, whether horizontally or vertically; you can use LinearLayout. Similarly, if you wish to display elements and views in rows and columns, you use the GridLayout.

Specification of Android GridLayout
1. Row and Column Specs

Using the row and column specs, you can specify how rows and columns are required and how the elements should be oriented. To do so, you can use the rowSpec and columnSpec layout parameters.

2. Default Cell Assignment
If the element doesn't specify where it needs to be present, then the GridLayout automatically assigns its position. The position of the component is based on its orientation, rowCount, and columnCount properties.

3. Space
To provide space between the elements of the grid, you can use the topMargin, bottomMargin, leftMargin, and rightMargin layout parameters.

Attributes of GridLayout in Android

Attribute Name	Description
android:columnCount	The "columnCount" attribute is used to specify the maximum number of columns you can attain while positioning the elements.
android:columnOrderPreserved	The "columnOrderPreserved" when true makes the column boundaries follow the column indices.
android:rowCount	The "rowCount" specifies the maximum number of rows you can attain while positioning the elements.
android:rowOrderPreserved	The "rowOrderPreserved", when set true, makes the row boundaries follow the row indices.
android:useDefaultMargins	The "useDefaultMargins" attribute is used to specify whether or not to use the default margins for your GridLayout. It has true or false values.

Methods involved in Android GridLayout
GridLayout has several public methods that allow us to manage and add functionalities to the layout.

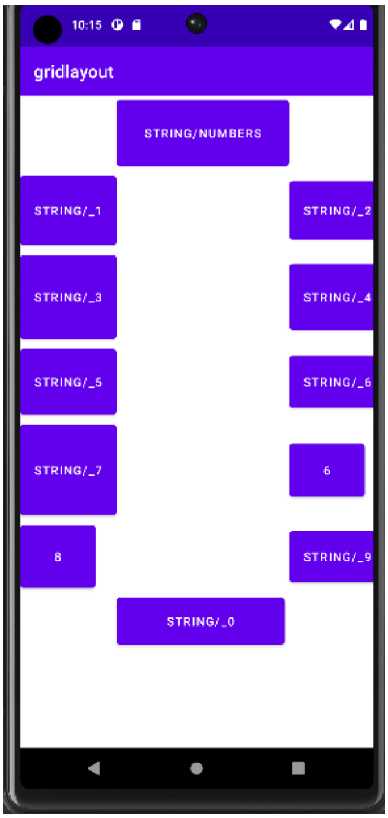
Method Name	Description
getAlignmentMode()	It is used to know the current alignment mode.
getColumnCount()	It is used to get the number of columns you can have at max while positioning elements.

getRowCount()	It is used to get the number of rows you can have at max while positioning elements.
setAlignmentMode(alignmentMode: Int)	Using the setAlignmentMode() you can set the alignment of your GridLayout whether top, left, right, center, etc.
setColumnCount(columnCount: Int)	It is used to set the number of columns you can have at max while positioning elements.
setColumnOrderPreserved (columnOrderPreserved: Boolean)	It is used to make the column boundaries follow the order of column indices.
setRowCount(rowCount: Int)	It is used to set the number of rows you can have at max while positioning elements.
setRowOrderPreserved(rowOrderPreserved: Boolean)	It is used to make the row boundaries follow the order of row indices.
setUseDefaultMargins(useDefaultMargins: Boolean)	It is used to specify that the GridLayout would use default margins or not.

ALGORITHM:

Conclusion:

GRID LAYOUT



java file

```
package com.example.myapplicationgrid;
import android.os.Bundle;

import com.google.android.material.snackbar.Snackbar;

    import androidx.appcompat.app.AppCompatActivity;

    import android.view.View;

    import android.view.Menu;
    import android.view.MenuItem;
    import android.widget.Button;

public class MainActivity extends AppCompatActivity {


  Button b1,b2,b3,b4,b5,b6,b7,b8,b9;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    b1=(Button) findViewById(R.id.button);
    b1=(Button) findViewById(R.id.button2);
    b2=(Button) findViewById(R.id.button3);
    b3=(Button) findViewById(R.id.button4);
    b4=(Button) findViewById(R.id.button5);
    b5=(Button) findViewById(R.id.button6);
    b6=(Button) findViewById(R.id.button7);
    b7=(Button) findViewById(R.id.button8);
    b8=(Button) findViewById(R.id.button9);
    b9=(Button) findViewById(R.id.button10);
    b1=(Button) findViewById(R.id.button11);
    b1=(Button) findViewById(R.id.button12);

  }
}
```

xml file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
```

```
   tools:context=".MainActivity">
   <androidx.gridlayout.widget.GridLayout android:id="@+id/gridLayout"
android:layout_width="match_parent" android:layout_height="match_parent">
      <Button android:id="@+id/button" android:layout_width="201dp"
android:layout_height="88dp" android:text="@string/numbers" app:layout_column="1"
app:layout_row="0"/>
      <Button android:id="@+id/button2" android:layout_width="wrap_content"
android:layout_height="92dp" android:text="@string/_1" app:layout_column="0"
app:layout_row="1"/>
      <Button android:id="@+id/button3" android:layout_width="wrap_content"
android:layout_height="79dp" android:text="@string/_2" app:layout_column="2"
app:layout_row="1"/>
      <Button android:id="@+id/button4" android:layout_width="wrap_content"
android:layout_height="109dp" android:text="@string/_3" app:layout_column="0"
app:layout_row="2"/>
      <Button android:id="@+id/button5" android:layout_width="wrap_content"
android:layout_height="88dp" android:text="@string/_4" app:layout_column="2"
app:layout_row="2"/>
      <Button android:id="@+id/button6" android:layout_width="wrap_content"
android:layout_height="88dp" android:text="@string/_5" app:layout_column="0"
app:layout_row="3"/>
      <Button android:id="@+id/button7" android:layout_width="wrap_content"
android:layout_height="72dp" android:text="@string/_6" app:layout_column="2"
app:layout_row="3"/>
      <Button android:id="@+id/button8" android:layout_width="wrap_content"
android:layout_height="117dp" android:text="@string/_7" app:layout_column="0"
app:layout_row="4"/>
      <Button android:id="@+id/button9" android:layout_width="wrap_content"
android:layout_height="73dp" android:text="6" app:layout_column="2" app:layout_row="4"
         tools:ignore="HardcodedText" />
      <Button android:id="@+id/button10" android:layout_width="wrap_content"
android:layout_height="84dp" android:text="8" app:layout_column="0" app:layout_row="5"
         tools:ignore="HardcodedText" />
      <Button android:id="@+id/button11" android:layout_width="wrap_content"
android:layout_height="71dp" android:text="@string/_9" app:layout_column="2"
app:layout_row="5"/>
      <Button android:id="@+id/button12" android:layout_width="196dp"
android:layout_height="66dp" android:text="@string/_0" app:layout_column="1"
app:layout_row="6"/>
   </androidx.gridlayout.widget.GridLayout>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```