第一類

# UI 設計及語法應用

101. 點餐系統

102. 計算 BMI 值

103. 動態密碼顯示

104. 顯示圖片

105. 全螢幕對話方塊

106. 驗證 Activity

107. 跑馬燈

108. 電影租片清單

109. 訊息通知

110. 自動收合的廣告版位

Android 9

# 104. 顯示圖片

## 題解說明

A.解題要項

- 熟悉按鈕(Button)的事件處理機制
- 了解 ImageView 的常用屬性與方法
- 熟悉 AlterDialog 的生成、顯示，以及按鈕事件處理
- 熟悉基本的 Toast 元件的使用
- 了解 Log 的使用時機

B.重點描述

1. 根據題本說明，專案內已經設計好佈局檔案(main.xml)；請直接開啓主程式 MainActivity.java，找到 onCreate(...)方法
2. 設定 mButton1 的事件註冊、並覆寫 onClick(...)方法，方法內部需要利用 AlertDialog.Builder(...)生成對話框物件，與 101 題目類似；不過，此題的對話框必須同時處理 positive 與 negative 按鈕的個別事件。以下為 mButton1 的主要架構：

```
mButton01.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        new AlertDialog.Builder(MainActivity.this)
            .setTitle("要顯示圖片嗎？")
            .setMessage("要顯示圖片嗎？")
            .setPositiveButton("YES", yesHandler)
            .setNegativeButton("NO", noHandler)
            .show();
    }
});
```

3. 以下列出 ImageView 圖片設定、顯示控制的參考建議：

   (1) 當 ImageView 不顯示圖片、且點擊後無動作，建議使用以下指令：

```
mImageView.setImageDrawable(null);
```

mImageView.setVisibility(View.GONE);

(2) 當切換 ImageView 圖片、而且是可見、可點擊，建議使用以下指令：

mImageView.setImageResource(R.drawable.csf);

mImageView.setVisibility(View.VISIBLE);

4. 設定 mButton2 的事件註冊、覆寫 onClick(...)方法，方法內部只需要利用上面隱藏圖片的指令，再搭配 Toast 顯示吐司條訊息。

5. 接著，根據題目要求，完成 ImageView 點擊事件處理的 onClick(...)方法實作：

Log.d("debug", "你點選了圖片");

Note 若程式可以正常編譯與安裝、但是卻無法正常執行，且在 Logcat 發現以下例外錯誤：
java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.
請修改 values/styles.xml，將 style 屬性 parent 改為以下內容值即可："Base.Theme.AppCompat.Light.DarkActionBar"

程式實作

**MainActivity.java**

```
1  package COM.TQC.GDD01;
2
3  import android.app.AlertDialog;
4  import android.content.Context;
5  import android.content.DialogInterface;
6  import android.graphics.Bitmap;
7  import android.graphics.BitmapFactory;
8  import android.os.Bundle;
9  import android.support.v7.app.AppCompatActivity;
10 import android.util.Log;
11 import android.view.View;
12 import android.view.View.OnClickListener;
13 import android.widget.Button;
```

```
14  import android.widget.ImageView;
15  import android.widget.Toast;
16
17
18  public class MainActivity extends AppCompatActivity {
19      private Button mButton01;
20      private Button mButton02;
21      private ImageView mImageView;
22      private final Context context = this;
23
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26        super.onCreate(savedInstanceState);
27        setContentView(R.layout.activity_main);
28        //context = getApplicationContext();
29        mImageView=(ImageView) findViewById(R.id.imageView);
30        //    TO DO  (1) 按下「圖片」按鈕，顯示一個「YES/NO對話視窗」，對話
    窗標題顯示【要顯示圖片嗎？】，對話內容顯示【選擇YES 會顯示圖片】，當按下【YES
    】顯示「R.drawable.tca」的檔案在ImageView當中。 當按下【NO】顯示，清空下
    方ImageView圖片。
31        mButton01=(Button) findViewById(R.id.button);
32        mButton01.setOnClickListener(new OnClickListener() {
33          @Override
34          public void onClick(View v) {
35             new AlertDialog.Builder(MainActivity.this)
36                .setTitle("要顯示圖片嗎？")
37                .setMessage("要顯示圖片嗎？")
38                .setPositiveButton("YES", new
39  DialogInterface.OnClickListener() {
                      @Override
40                    public void onClick(DialogInterface dialog, int
    which) {
41                        mImageView.setImageResource(R.drawable.csf);
42                        mImageView.setVisibility(View.VISIBLE);
43                    }
44                 })
45                .setNegativeButton("NO", new
    DialogInterface.OnClickListener() {
46                    @Override
```

```
47                      public void onClick(DialogInterface dialog, int
    which) {
48                              mImageView.setImageDrawable(null);
49                              mImageView.setVisibility(View.GONE);
50                                }
51                  })
52                 .show();
53              }
54          });
55          //    TO DO  (2) 當按下「清除」按鈕，清空下方ImageView圖片，並用
    Toast訊息方式於畫面中顯示【已經清除圖片】文字。
56          mButton02=(Button) findViewById(R.id.button2);
57          mButton02.setOnClickListener(new OnClickListener() {
58            @Override
59            public void onClick(View v) {
60                mImageView.setImageDrawable(null);
61                mImageView.setVisibility(View.GONE);
62                Toast.makeText( MainActivity.this, "已經清除圖片",
    Toast.LENGTH_LONG).show();
63            }
64          });
65          //   TO DO  (3) 當ImageView有圖片時，點選該圖會以Log.d訊息方式設
    定tag為debug並顯示【你點選了圖片】。
66          mImageView.setOnClickListener(new OnClickListener() {
67                @Override
68                public void onClick(View v) {
69                    Log.d("debug", "你點選了圖片");
70                }
71          });
72
73      }
74  }
```

## 程式說明

| 行　　數 | 說　　明 |
|---|---|
| 32~54 | 設定「顯示圖片」按鈕(mButton1)的事件註冊、並撰寫相關處理程式；最主要是：生成對話框物件，隨即設定標題，訊息內容，以及兩個按鈕的事件處理，最後顯示出該對話框 |

| | |
|---|---|
| 38~44 | 利用 setPositiveButton(...)設定「YES」按鈕被點擊的事件處理程式；主要是覆寫 onClick(...)方法以顯示圖片 |
| 45~51 | 利用 setNegativeButton(...) 設定「NO」按鈕被點擊的事件處理程式；主要是覆寫 onClick(...)方法以隱藏圖片 |
| 57~64 | 設定「清除」按鈕(mButton2) 的事件註冊與撰寫相關處理程式 |
| 69 | 根據題意，當圖片被點擊後利用 Log.d (debug) 顯示相關訊息 |

第二類

# 資料儲存、解析與交換

# 206. 姓名清單

## 題解說明

A.解題要項

- 能利用 ResourceManager 讀取事先定義的資源陣列（字串陣列）
- 能利用 SQLiteDatabase 控制物件管理 SQLite 資料庫、並利用 SQL 指令建立新資料表
- 能利用指令處理資料表內容：新增、查詢、修改、刪除(CRUD)
- 能利用 Cursor 存取資料表內的記錄、並讀取各類型的欄位資料
- 能利用 ArrayAdapter 與 Spinner 顯示動態資料

B.重點描述

1. 需要了解 SQLite 資料庫基本觀念、SQL 語法，並能利用 SQLiteDatabase 進行資料庫與資料表的處理。基本觀念與題目 205 的解題觀念類似，建議參考題目 205 重點描述的步驟 1 內文。
2. 根據題意說明，此題的主畫面(/res/layout/main.xml)已經設計好；而程式內所需要的選項清單已經被定義在 /res/values/strings.xml 檔案內，是以字串陣列的形式存在(id:strNames)，請開啓並觀察檔案內容，程式中需要利用 ResourceManager 來讀取該字串陣列的內容。
3. 開啓 GDD02.java 程式，可以發現其內部已經定義了四個字串常數（DBNAME、TABLENAME、FIELD01_NAME、FIELD02_NAME），主要是作為資料庫名稱、資料表名稱以及欄位名稱使用；此外，程式內也定義了幾個成員變數（dataBase、cursor、Spinner01、strNames、orderNames、recordCount），也請讀者順便觀察這些成員變數。接著，依序完成以下步驟：

   (1) 利用 ResourceManager 從資源檔中取得字串陣列(strNames)。並將字串指定給 strNames：

   ```
   strNames = resources.getStringArray(R.array.strNames);
   ```

(2) 利用 openOrCreateDatabase(...)取得資料庫控制物件，並利用程式內給定的 SQL 指令建立程式內所需要的資料表：

```
dataBase =
        openOrCreateDatabase( DBNAME, MODE_PRIVATE, null);
dataBase.execSQL( CREATE_SQL );
```

(3) 利用 SQL 指令來清空資料表的內容，注意 **from 後方有一個空格**：

```
dataBase.execSQL("delete from " + TABLENAME );
```

(4) 利用迴圈將陣列元素逐一新增到資料表內，此時我們利用 String.format(...) 方法來產生特定格式的 SQL 指令，務必注意指令內的單引號與空格的完整性，以避免產生 SQL 語法錯誤：

```
for (int i=0; i<strNames.length; i++) {
    String sql1 = String.format("insert into %s(%s) values('%s')",
                TABLENAME, FIELD02_NAME, strNames[i]);
    dataBase.execSQL(sql1);
}
```

(5) 接著，先利用 SQL 指令從資料表中讀出排序好的資料(order by)、再利用 Cursor 遍訪所有記錄、取出必要欄位內容（編號: 1）並儲存到字串陣列 orderNames；因題目要求資料顯示時必須由小到大排序好，因此，必須利用 SQL 的 order by XXX 參數來進行排序：

```
String sql2 = String.format("select * from %s ORDER BY %s",
            TABLENAME, FIELD02_NAME);
cursor = dataBase.rawQuery( sql2, null);
recordCount = cursor.getCount();
orderNames = new String[recordCount]; //  動態配置陣列空間
cursor.moveToFirst();
for (int i=0; i<recordCount; i++) {
    orderNames[i] = cursor.getString(1); //  第二個欄位:編號為 1
```

```
        cursor.moveToNext();
    }
```

(6) 因為程式內部已有生成 ArrayAdapter 物件的指令，所以，只需要將 adapter 指定給 Spinner 做為資料來源即可。

程式實作

GDD02.java

```
1  package com.tqc.gdd02;
2
3  import android.app.Activity;
4  import android.content.res.Resources;
5  import android.database.Cursor;
6  import android.database.sqlite.SQLiteDatabase;
7  import android.os.Bundle;
8  import android.widget.ArrayAdapter;
9  import android.widget.Spinner;
10
11 public class GDD02 extends Activity
12 {
13   private static final String DBNAME = "MY_DB";
14   private static final String TABLENAME = "MY_TABLE";
15   private static final String FIELD01_NAME = "_id";
16   private static final String FIELD02_NAME = "_text1";
17   private SQLiteDatabase dataBase;
18   private Spinner Spinner01;
19   private CharSequence[] strNames;
20   private Cursor cursor;
21   private String orderNames[];
22   //資料筆數
23   private int recordCount;
24
25   /**
26    * Called when the activity is first created.
27    */
28   @Override
29   public void onCreate(Bundle savedInstanceState)
```

```
30    {
31      super.onCreate(savedInstanceState);
32      setContentView(R.layout.main);
33
34      Spinner01 = (Spinner) findViewById(R.id.Spinner01);
35      Resources resources = getResources();
36      String CREATE_SQL = "create table if not exists " + TABLENAME + "
    (" + FIELD01_NAME + " integer primary key autoincrement, " + FIELD02_NAME
    + " varchar not null);";
37
38      //TO DO
39      // 1. 從資源檔中取得字串陣列, 並將資料逐一新增到資料表中
40      strNames = resources.getStringArray(R.array.strNames);
41      // 2. 建立資料庫與資料表
42      dataBase = openOrCreateDatabase( DBNAME, MODE_PRIVATE, null);
43      dataBase.execSQL( CREATE_SQL );
44      // 3. 根據題意, 每次執行都先清空資料表。
45      dataBase.execSQL("delete from " + TABLENAME );
46      // 4. 逐一將字串陣列的資料項目新增到資料表內
47      for (int i=0; i<strNames.length; i++) {
48        String sql1 = String.format("insert into %s(%s) values('%s')",
49                TABLENAME, FIELD02_NAME, strNames[i]);
50        dataBase.execSQL(sql1);
51      }
52      // 5. 再從資料表中取出資料(已排序)、並存到字串陣列 orderNames
53      String sql2 = String.format("select * from %s ORDER BY %s",
54              TABLENAME, FIELD02_NAME);
55      cursor = dataBase.rawQuery( sql2, null);
56      recordCount = cursor.getCount();
57      orderNames = new String[recordCount];
58      //
59      for (int i=0; i<recordCount; i++) {
60        cursor.moveToPosition(i);
61        orderNames[i] = cursor.getString(1); // 第二個欄位:編號為 1
62      }
63
64      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, orderNames);
65
```

```
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
    pdown_item);
66
67      //TO DO Spinner
68      Spinner01.setAdapter(adapter);
69    }
70  }
```

## 程式說明

| 行　　數 | 說　　明 |
|---|---|
| 40 | 利用 ResourceManager 控制物件從資源檔中取得字串陣列 |
| 42 | 利用 openOrCreateDatabase(...)生成 SQLiteDatabase 控制物件 db |
| 43 | 利用 db.execSQL("create table ...")建立資料表 MY_TABLE |
| 45 | 利用 db.execSQL("delete from MY_TABLE")清除舊有的資料；請注意：字串最後的空格不可短少 |
| 47~51 | 利用迴圈，將陣列元素逐一新增到資料表內 |
| 53~55 | 利用 db.rawQuery("select * from ... ORDER BY ...") 取得 Cursor 物件以讀取資料表 MY_TABLE 的記錄；此時必須利用 ORDER BY XXX 來取得排序好的記錄清單 |
| 56~57 | 根據 Cursor 記錄的數量生成陣列物件 orderNames |
| 59~62 | 利用迴圈逐一取得每筆記錄的必要欄位資料，並儲存到陣列 orderNames 內；以利後續 Spinner 顯示出已排序好的資料。 |
| 68 | 將 adapter 指定給 Spinner 物件作為資料顯示來源 |

第三類

# 服務應用

301. MP3 播放器

302. 畫布程式

303. 程式背景音樂

304. 手機網路流量統計

305. 影片播放器

306. 接收 SMS

307. PIP 子母畫面模式

308. 經緯度查/反查地址

309. GPS 地標權限

310. 台北市運動中心網路連線 API 解析

# 302. 畫布程式

## 題解說明

A.解題要項

- 熟悉 SurfaceView 與 SurfaceHolder 的基本觀念與使用步驟
- 能利用 Window/WindowManager 切換全螢幕的顯示方式
- 能利用 GestureDetector 與 GestureDetector.OnGestureListener 介面偵測與處理使用者的手勢動作
- 了解 View.OnTouchListener 的事件處理機制
- 能利用 Canvas（畫布）、Color（顏色）、Paint（畫筆）進行繪圖處理

B.重點描述

1. 此題需要利用 SurfaceView 進行繪圖，其基本使用步驟如下：

   (1) 先取得 SurfaceView 物件的控制器(SurfaceHolder)：

   ```
   mSurfaceView01 = findViewById(R.id.surfaceView);
   mSurfaceHolder01 = mSurfaceView01.getHolder();
   ```

   (2) 利用 SurfaceHolder 鎖定全部區域(或部分區域)取得繪圖控制物件：

   ```
   Canvas canvas = mSurfaceHolder01.lockCanvas();
   ```

   (3) 利用 Canvas 物件進行繪圖指令的執行，此時還需要知道兩個類別：Color（顏色）、Paint（畫筆）。Color 用來產生繪圖顏色的物件；Paint 則是可以設定繪圖指令的顏色、樣式、線條粗細...等等。

   (4) 最後，釋放繪圖區域的鎖定：

   ```
   mSurfaceHolder01.unlockCanvasAndPost(canvas);
   ```

Note 若需要進行遊戲或動畫繪製，可能需要搭配實作 SurfaceHolder.Callback 介面以及執行緒來進行多工處理。詳細資訊可以參考 107 解題說明。

2. 觀察佈局檔 main.xml，發覺內部只有 LinearLayout，故須手動置入 SurfaceView 元件，並完成以下三個參數的設定：

| 元件 | 屬性 | 屬性值 | 註解 |
|---|---|---|---|
| SurfaceView | id | surfaceView | |
| | layout_width | match_parent | |
| | layout_height | match_parent | |

Note 雖然題目敘述中有提到【背景為黑色】，但是，並不需要將 SurfaceView 的背景色設定為【黑色】；否則程式執行時會出現問題。

3. 此題需要先切換為全螢幕，請開啓 GDD03.java 程式，在 **setContentView(...) 方法之前**輸入以下指令即可將畫面切換為全螢幕；主要是利用 Window 控制物件(win)與 WindowManager 類別參數：

```
win.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
this.requestWindowFeature(Window.FEATURE_NO_TITLE);
```

Note 上述設定全螢幕的指令必須在 setContentView(...) 之前完成，否則會沒有作用（無效）。

4. 在 setContentView(...)方法之後輸入以下指令以便生成手勢偵測(GestureDetector)物件：

```
detector = new GestureDetector(GDD03.this, GDD03.this);
```

接著讓 GDD03 實作 GestureDetector.OnGestureListener 介面內的相關方法，其中 **onDown**(...) 即是手勢偵測為【按下】事件時會被觸發的方法，稍後我們會改寫其內部的處理指令（繪圖）。

5. 根據題意要求，當發生**點擊**動作時必須在手指點擊的位置繪製出一個半徑 30 的實心圓形。此時需要搭配 SurfaceView，請在前一步驟指令的下方，依序輸入以下步驟內的相關指令：

(1) 先取得 SurfaceView 物件的控制器(SurfaceHolder)：

```
mSurfaceView01 = findViewById(R.id.surfaceView);
```

```
mSurfaceHolder01 = mSurfaceView01.getHolder();
```

(2) 接著呼叫 SurfaceView 的 setOnTouchListener(GDD03.this)來設定事件處理：

```
mSurfaceView01.setOnTouchListener(GDD03.this);
```

同時必須讓 GDD03 實作 View.OnTouchListener 介面內的抽象方法；並覆寫其中的 **onTouch**(...) 方法內：呼叫以下指令將碰觸事件資訊傳送給 GestureDetector 物件以進行手勢判讀：

```
return detector.onTouchEvent(event);
```

注意：此題 onTouch(...) 方法內部已經寫好，無需再修改。

Note 此外，也可以覆寫 GDD03 Activity 的 onTouchEvent(...) 來取代 SurfaceView 的 OnTouchListener 內的 onTouch(...) 方法。

6. 最後，找到 onDown(MotionEvent e)方法；利用 SurfaceHolder 進行畫布的鎖定、並由 MotionEvent 物件取得點擊事件的所在座標(x,y)、進行繪圖，最後，解除繪圖區的鎖定：

```
// 鎖定畫布
Canvas canvas = mSurfaceHolder01.lockCanvas();
// 清除整個畫布為黑色!
canvas.drawColor(Color.BLACK);
// 改變畫筆顏色後畫出實心圓形!
Paint paint = new Paint();
paint.setColor(Color.WHITE);
int x = (int) e.getX();
int y = (int) e.getY();
canvas.drawCircle( x, y, 30, paint);
mSurfaceHolder01.unlockCanvasAndPost(canvas);
return true;
```

程式實作

/res/layout/main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3    xmlns:android="http://schemas.android.com/apk/res/android"
4    android:background="@color/white"
5    android:orientation="vertical"
6    android:layout_width="fill_parent"
7    android:layout_height="fill_parent">
8
9    <SurfaceView
10         android:id="@+id/surfaceView"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent" />
13
14 </LinearLayout>
```

## 程式說明

| 行　　數 | 說　　明 |
|---|---|
| 9~12 | 加入 SurfaceView，設定 width,heigh 為"match_parent"，並確認其 id 設定值(surfaceView) |

GDD03.java

```
1  package com.tqc.gdd03;
2
3  import android.app.Activity;
4  import android.graphics.Canvas;
5  import android.graphics.Color;
6  import android.graphics.Paint;
7  import android.os.Bundle;
8  import android.view.GestureDetector;
9  import android.view.MotionEvent;
10 import android.view.SurfaceHolder;
11 import android.view.SurfaceView;
12 import android.view.View;
13 import android.view.Window;
```

```
14  import android.view.WindowManager;
15
16  public class GDD03 extends Activity implements
    GestureDetector.OnGestureListener, View.OnTouchListener {
17    public static SurfaceView mSurfaceView01;
18    public static SurfaceHolder mSurfaceHolder01;
19    public GestureDetector detector;
20
21    /** Called when the activity is first created. */
22    @Override
23    public void onCreate(Bundle savedInstanceState)
24    {
25      super.onCreate(savedInstanceState);
26      Window win = getWindow();
27      // TO DO 程式一執行即進入全螢幕模式，沒有狀態列
28      win.setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
    WindowManager.LayoutParams.FLAG_FULLSCREEN);
29      requestWindowFeature(Window.FEATURE_NO_TITLE);
30      //
31      setContentView(R.layout.main);
32
33      // TO DO
34      detector = new GestureDetector(GDD03.this,GDD03.this);
35      mSurfaceView01 = findViewById(R.id.surfaceView);
36      mSurfaceHolder01 = mSurfaceView01.getHolder();
37      mSurfaceView01.setOnTouchListener(GDD03.this);
38    }
39
40    @Override
41    public boolean onDown(MotionEvent e) {
42     //TO DO
43      // 鎖定畫布
44      Canvas canvas = mSurfaceHolder01.lockCanvas();
45      // 清除整個畫布為黑色!
46      canvas.drawColor(Color.BLACK);
47      // 改變畫筆顏色後畫出實心圓形!
48      Paint paint = new Paint();
49      paint.setColor(Color.WHITE);
50      int x = (int) e.getX();
```

【302. 畫布程式】

```
51      int y = (int) e.getY();
52      canvas.drawCircle( x, y, 30, paint);
53      mSurfaceHolder01.unlockCanvasAndPost(canvas);
54      //
55      return true;
56    }
57
58    @Override
59    public void onShowPress(MotionEvent e) {
60
61    }
62
63    @Override
64    public boolean onSingleTapUp(MotionEvent e) {
65      return false;
66    }
67
68    @Override
69    public boolean onScroll(MotionEvent e1, MotionEvent e2, float
  distanceX, float distanceY) {
70      return false;
71    }
72
73    @Override
74    public void onLongPress(MotionEvent e) {
75
76    }
77
78    @Override
79    public boolean onFling(MotionEvent e1, MotionEvent e2, float
  velocityX, float velocityY) {
80      return false;
81    }
82
83    @Override
84    public boolean onTouch(View v, MotionEvent event) {
85      return detector.onTouchEvent(event);
86    }
87
```

```
88    @Override
89    public void onPointerCaptureChanged(boolean hasCapture) {
90
91    }
92  }
```

## 程式說明

| 行　　數 | 說　　明 |
|---|---|
| 16 | 讓 GDD03 實作介面 View.OnTouchListener,GestureDetector.OnGestureListener |
| 28~29 | 利用 Window 控制物件、結合 WindowManager 類別參數將畫面切換為全螢幕 |
| 34 | 生成 GestureDetector 手勢偵測物件，此時 GDD03 必須實作 GestureDetector.OnGestureListener 介面內的抽象方法：onDown(...), onShowPress(...), onSingleTapUp(...), onScroll(...), onLongPress(...), onFling(...) 等抽象方法 |
| 35 | 利用 findViewById(...) 綁定 SurfaceView 元件 |
| 36 | 取得 SurfaceView 的 SurfaceHolder 控制物件 |
| 37 | 進行 SurfaceView 的 onTouch 事件註冊，此時 GDD03 必須實作 View.OnTouchListener 介面內的抽象方法 |
| 43~53 | 改寫 onDown(...) 方法；利用 SurfaceHolder 進行畫布的鎖定、取得點擊事件的所在座標(x,y)、進行繪圖，最後，解除繪圖區的鎖定 |