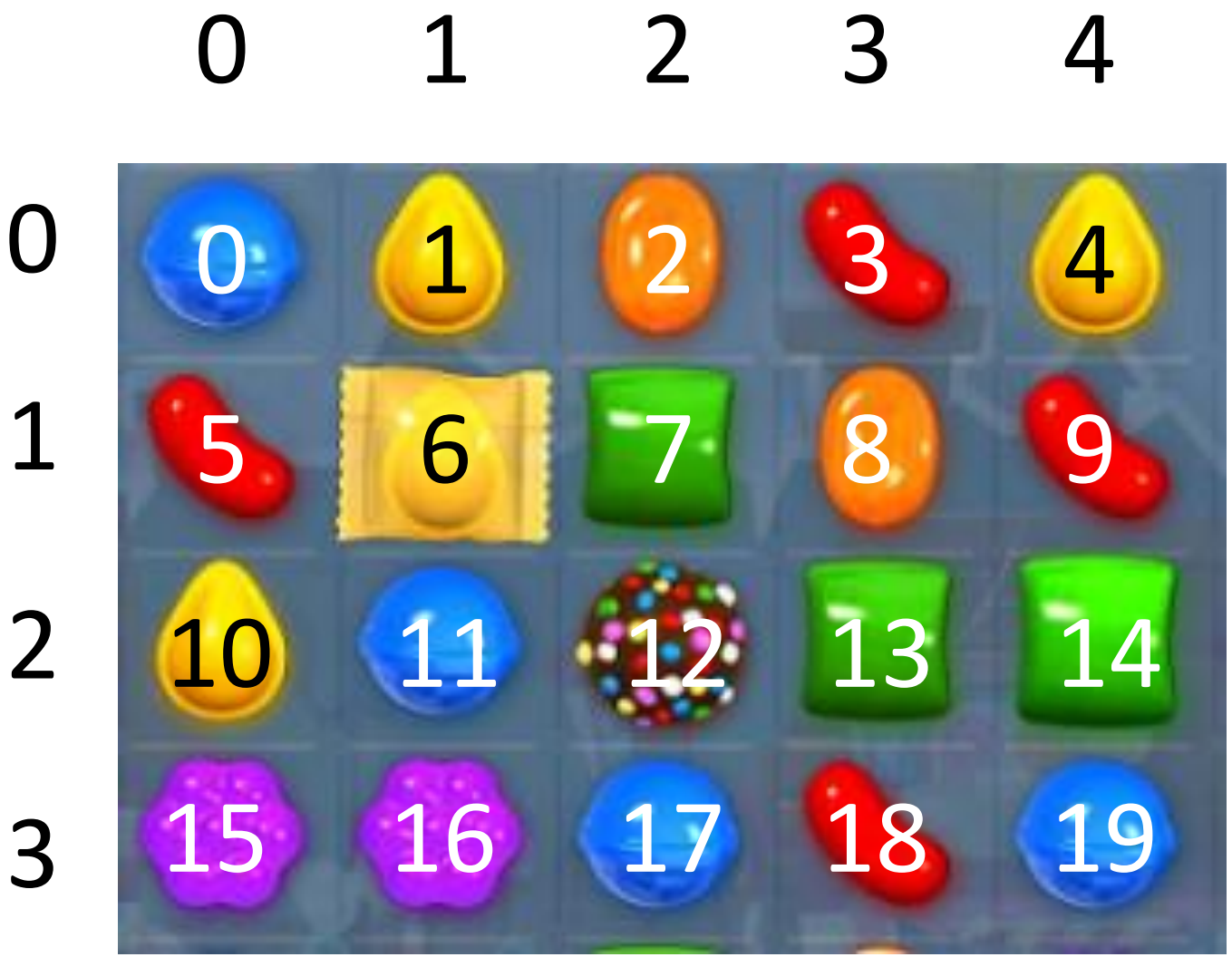
# Arrays of Buttons

Inside Android


# 2D Arrays

[row][col]

What colour of candy is in [2][1]?

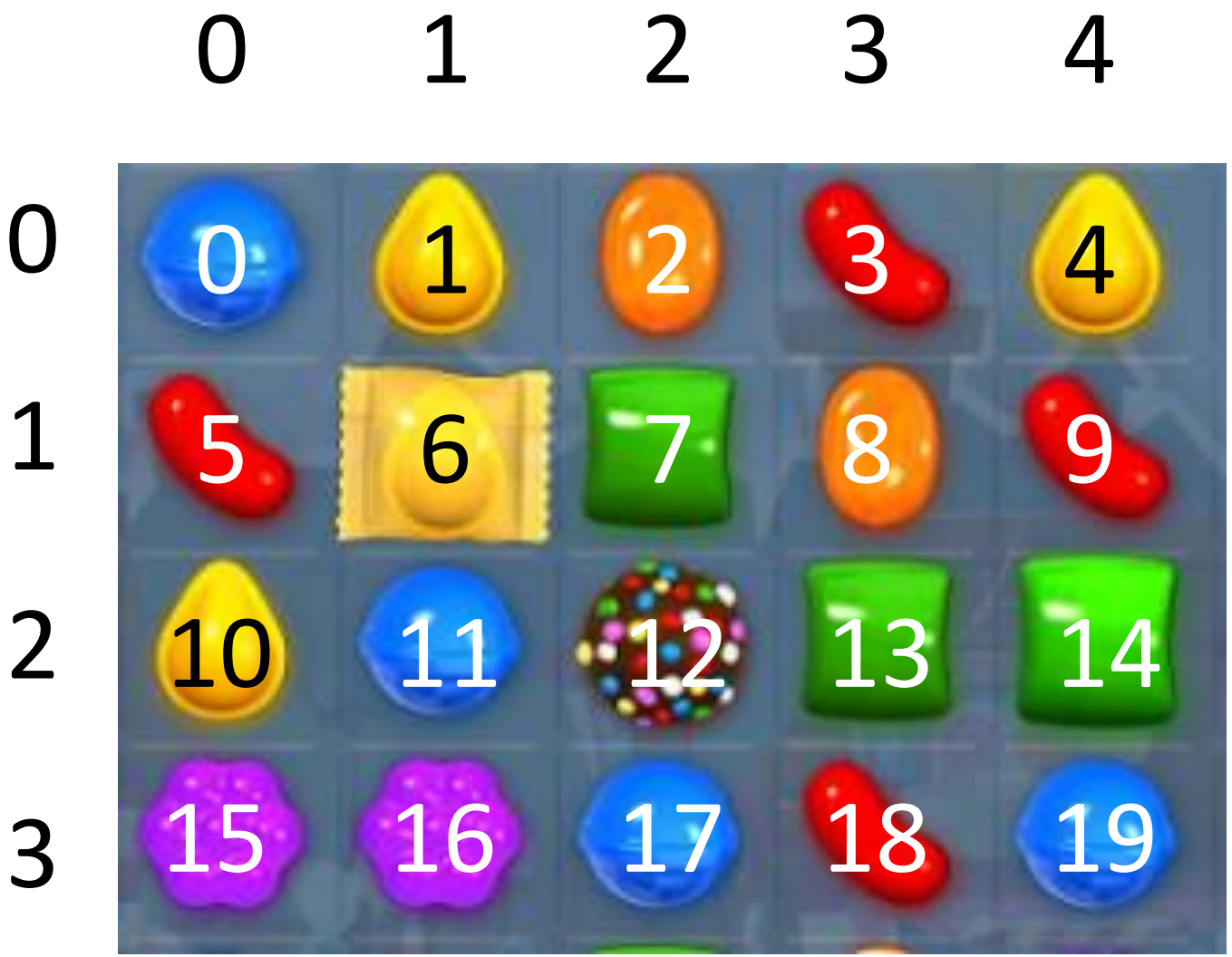


[row][col]

[2] = row,
[1] = col.
Blue.

[row][col]

What colour of candy is in [3][1]?



[row][col]

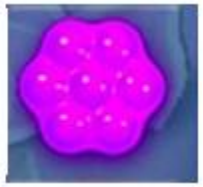
[3] = row,
[1] = col.
Purple.

I have these images:


0
oval.gif


1
sq.gif


2
round.gif

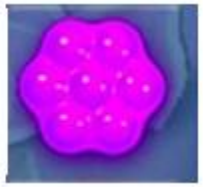

3
drop.gif

I have these images:


0
oval.gif


1
sq.gif


2
round.gif


3
drop.gif

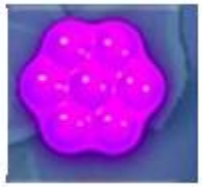
I want these Image Views:

I have these images:


0
oval.gif


1
sq.gif


2
round.gif


3
drop.gif

I want these Image Views:

The Tracking Array is like this:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

# In Android, it is easier to make an ImageView array 1D.

In Android, it is easier to make an ImageView array 1D.


Then, we use a gridLayout to make it APPEAR as a 2D array.

# Our 1D ImageView array:


Appears like:

The tracking array is 2D.

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

How many elements are in the 1D ImageView array?

# Our 1D ImageView array:


Appears like:

The tracking array is 2D.

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

How many elements are in the 1D ImageView array?

15

# Our 1D ImageView array:


## Appears like:

## The tracking array is 2D.

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

In the 2D tracking array, how many rows and columns?

# Our 1D ImageView array:


Appears like:

The tracking array is 2D.

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

In the 2D tracking array, how many rows and columns?

5 rows, 3 cols.

It is no accident that 5 * 3 = 15

Moving from 2D to 1D

# The IDs on the ImageView array match their index.


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [1][2] on the 2D array?

Moving from 2D to 1D

# The IDs on the ImageView array match their index.


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [1][2] on the 2D array?

# The IDs on the ImageView array match their index.

Moving from 2D to 1D


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [1][2] on the 2D array?

5

1*col + 2

Moving from 2D to 1D

# The IDs on the ImageView array match their index.


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [4][1] on the 2D array?

Moving from 2D to 1D

# The IDs on the ImageView array match their index.


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [4][1] on the 2D array?

Moving from 2D to 1D

# The IDs on the ImageView array match their index.


## The IDs are:


## The tracking array:

| | [0] | [1] | [2] |
|---|---|---|---|
| [0] | 1 | 2 | 0 |
| [1] | 1 | 1 | 2 |
| [2] | 2 | 2 | 1 |
| [3] | 1 | 2 | 3 |
| [4] | 1 | 0 | 0 |

What is the id of the 1D ImageArray what corresponds with [4][1] on the 2D array?

13

4*col + 1

Try this part of the sheet.

# Grid Array Algorithms

3.3 K

Name: _____

## 1D to 2D Array

0. ImageView arrays are only 1D so their IDs are unique. The 1D array for the grid below is like this:


However, if you look at the screen, it is laid out in a grid, or 2D array. We use a 2D int array to track these positions.



For the 1D ImageView array:

(a) How many elements?

.....

(b) What is the ID of the first orange oval?

.....

For the 2D int tracking array:

(c) How many rows?

.....

(d) How many columns?

.....

The references are from the 2D int tracking array. Write the corresponding ID of the 1D ImageView array.

(e) [0][0] .....

(f) [2][1] .....

(g) [3][2] .....

(h) [2][3] .....

(i) [1][3] .....

(j) [0][2] .....

```
package ca.gorskicompsci.www.mediumflowfree;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.GridLayout;
import android.widget.ImageView;

public class Game extends AppCompatActivity {

    int sol [][] ={{2, 6, 6, 1}, {7, 1, 3, 8}, {7, 7, 7, 8}, {4, 3, 8, 8}, {9, 9, 9, 4}};
    int cur[][] = {{2, 0, 0, 1}, {0, 1, 3, 0}, {0, 0, 2, 0}, {4, 3, 0, 0}, {0, 0, 0, 4}};


    int row = 5;
    int col = 4;


    ImageView pics[]=new ImageView[row*col];
```

Don't mess with the top classes and items!

Set up your row and column variables.

2D int tracking answer and solution arrays

1D Widget Array

We will need to link our code to the XML.

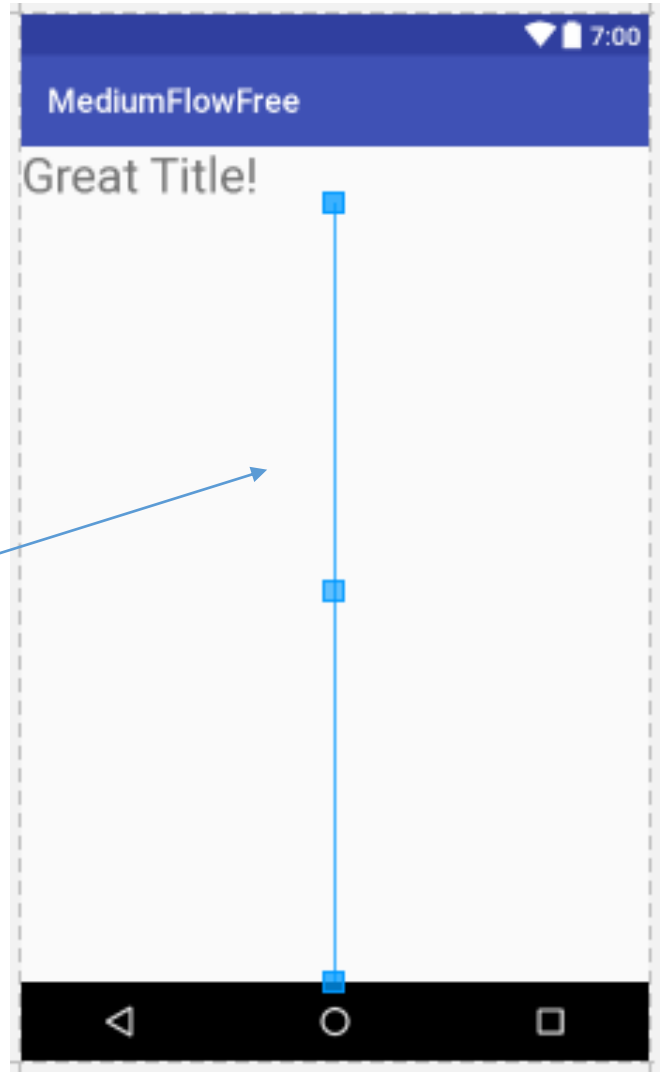
This involves making a blank grid array.

```
<?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="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Great Title!"
        android:textSize="30sp"/>

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_height="match_parent"
        android:rowCount="5"
        android:columnCount="4"
        android:id="@+id/griddy">
    </GridLayout>

</LinearLayout>
```




To hold the image array later on. It's empty now.

# The Complete Code Listing.

Be careful about cutting and pasting.

```xml
<?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="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Great Title!"
        android:textSize="30sp"/>

    <GridLayout
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_height="match_parent"
        android:rowCount="5"
        android:columnCount="4"
        android:id="@+id/mygrid">
    </GridLayout>

</LinearLayout>
```

```java
public class Game extends AppCompatActivity {
    int cur[][] = {{2, 0, 0, 1}, {0, 1, 3, 0}, {0, 0, 2, 0}, {4, 3, 0, 0}, {0, 0, 0, 4}};
    int row = 5;  int col = 4;
    ImageView pics[]=new ImageView[row*col];

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_game);
        GridLayout g = (GridLayout) findViewById(R.id.griddy);
        int m=0;
        for(int i=0; i<row; i++){
            for(int j=0; j<col; j++){
                pics[m]=new ImageView(this);
                setpic(pics[m], m);
                pics[m].setId(m);
                g.addView(pics[m]);
                m++;
            }
        }
    }
    public void setpic(ImageView i, int pos){
        int x = pos/col;
        int y = pos%col;
        int picnum = cur[x][y];
        if(picnum==1)
            i.setImageResource(R.drawable.bend);
        else if(picnum==2)
            i.setImageResource(R.drawable.rend);
    }
}
```

The following code makes the 1D button array.

It puts it in the Grid we created in the XML.

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

Import your grid.

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

Import your grid.

For 1D ImageArray

Loop through your 2D Tracking array.

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

Import your grid.

For 1D ImageArray

Loop through your 2D Tracking array.

New or Construct each Image

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

Import your grid.

For 1D ImageArray

Loop through your 2D Tracking array.

New or Construct each Image

Match the 1D Image Array to the 2D Tracking Array. Show it up

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

    GridLayout g = (GridLayout) findViewById(R.id.griddy);

    int m=0;
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            pics[m]=new ImageView(this);
            setpicStart(pics[m], m);
            pics[m].setId(m);
            g.addView(pics[m]);
            m++;
        }
    }
}
```

Don’t touch. Already there.

Import your grid.

For 1D ImageArray

Loop through your 2D Tracking array.

New or Construct each Image

Match the 1D Image Array to the 2D Tracking Array. Show it up

Add this element of the 1D image Array to the grid.

This section is the same as yesterday

(k) Using the numbers above, fill in the code for this game of candy crush.

```
public class MainActivity extends AppCompatActivity {

    int candy[][] = {{__, __, __, __}, {__, __, __, __}, {__, __, __, __}};
    int row = ___;
    int col = ___;
    ImageView pics[] = new ImageView[row * col];

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   GridLayout g = (GridLayout) findViewById(R.id.grid);
   int m = 0;
   for (int i = 0; i < row; i++) {
     for (int j = 0; j < col; j++) {
        pics[m] = new ImageView(this);
        setpic(pics[m], m);
        pics[m].setId(m);
        g.addView(pics[m]);
        m++;
     }
   }
 }
```

```
public void setpic(ImageView i, int pos) {
   int x = pos / col;
   int y = pos % col;
   int picnum = __________[x][y];
   if (picnum == ____)
      i.setImageResource(R.drawable._____);
   else if (picnum == _____)
      i.setImageResource(R.drawable._____);
   else if (picnum == _____)
      i.setImageResource(R.drawable._____);
   else
      i.setImageResource(R.drawable._____);
}
```

Converting between 1D to 2D is a little harder.

2Dim's x,y

1Dim's ID

```
public void setpic(ImageView i, int pos){
        int x = pos/col;
        int y = pos%col;
        int picnum = cur[x][y];
        if(picnum==1)
            i.setImageResource(R.drawable.bend);
        else if(picnum==2)
            i.setImageResource(R.drawable.rend);
    }
```

# Another example:

The image array is 1D.
It's actionCommands (or IDs in Android) will be 0-19.

The int array to track the screen will be 2D.
This will allow easy manipulation behind the scenes.

Conversion between 1D and 2D arrays is easy.

[row][col]
[2][1] = blue candy

Conversion between 1D and 2D arrays is easy.

[row][col]
[2][1] = blue candy

pos / col = x
11 / 5 = 2

pos % col = y
11 % 5 = 1

[row][col]
[2][3] = green candy

[row][col]
[2][3] = green candy

[row][col]
[2][3] = green candy

pos / col = x
13 / 5 = 2

pos % col = y
13 % 5 = 3

[row][col]
[3][1] = purple candy

[row][col]
[3][1] = purple candy

pos / col = x
16 / 5 = 3

pos % col = y
16 % 5 = 1

Try this section now.

If n is the ID of the ImageView:

```
int x = n / col;
int y = n % col;
```

1. Assume that you have an grid that is 6 (rows) x 5 (cols).

(a) How many ImageViews do you need? . . . . . .

(b) Given the ImageView IDs, determine each button's (x, y), aka (row, col), position in the int tracking array.

| | | | |
|---|---|---|---|
| 6 | 10 | 17 | 29 |