**CSC 4992 – Android App Development**

**Homework 2 – [your name]**

**50 points – Due March 4, 11am**

**Late deadline is March 6, 11:59pm, but 20% off**

**a)** Save this document with your name and the homework number somewhere in the file name.

**b)** Paste your code and screenshots into the document.

**c)** Gather the following files into a ZIP file with your name and the homework number somewhere in the file name:

● This document

● A zip file of your Android Studio project

● Any other optional files specific to your application

**d)** Submit the ZIP file to Canvas where you downloaded this document.

You've been hired by *Tic Tac Tracers* to write an Android application that plays tic tac toe. The game pits the user versus the computer and the first player to get their image three tiles across, down, or diagonal wins the game. The human player always goes first. If neither player gets three tiles in a row and all tiles are occupied, then the game is a tie.

Create the application with the following API levels:

● Minimum of API 24.

● Target of API 29 (this will be the default with the latest Android Studio version).

Using the following Android virtual device to develop and run your application:

● Nexus 6P API 24.

**Icon**

Create a custom icon for the application.

**Main screen**

The application has a **Main screen** containing the following controls:

**Toolbar**

A toolbar showing the application (custom) icon, application name, and an overflow menu. The overflow menu provides two options:

● **Settings** which opens the settings screen defined below.

● **About** which displays a one-button dialog box showing the application name and version, and author name. Use class **AlertDialog.Builder** and **this** for the context.

**Status section**

This is a label that shows the result of a game. Initially, it says message:

(game result)

After a game, it shows one of the following messages:

● HUMAN WINS!

● COMPUTER WINS!

● NOBODY WINS - TIE GAME!

**Board section**

The board consists of a three-by-three grid of tiles. The tiles could be implemented as image buttons. There are three settings that control its appearance:

● A background color for the board.

● A background color for the tiles.

● A pair of images representing the human and computer players.

Each of these settings is controlled by the settings screen defined below. The board also indicates whose turn is next.

**Stats section**

The stats section has the following three rows of controls:

● A label and read-only text box showing the number of games played.

● A label and read-only text boxes showing the number of human wins and the percentage of games won.

● A label and read-only text boxes showing the number of computer wins and the percentage of games won.

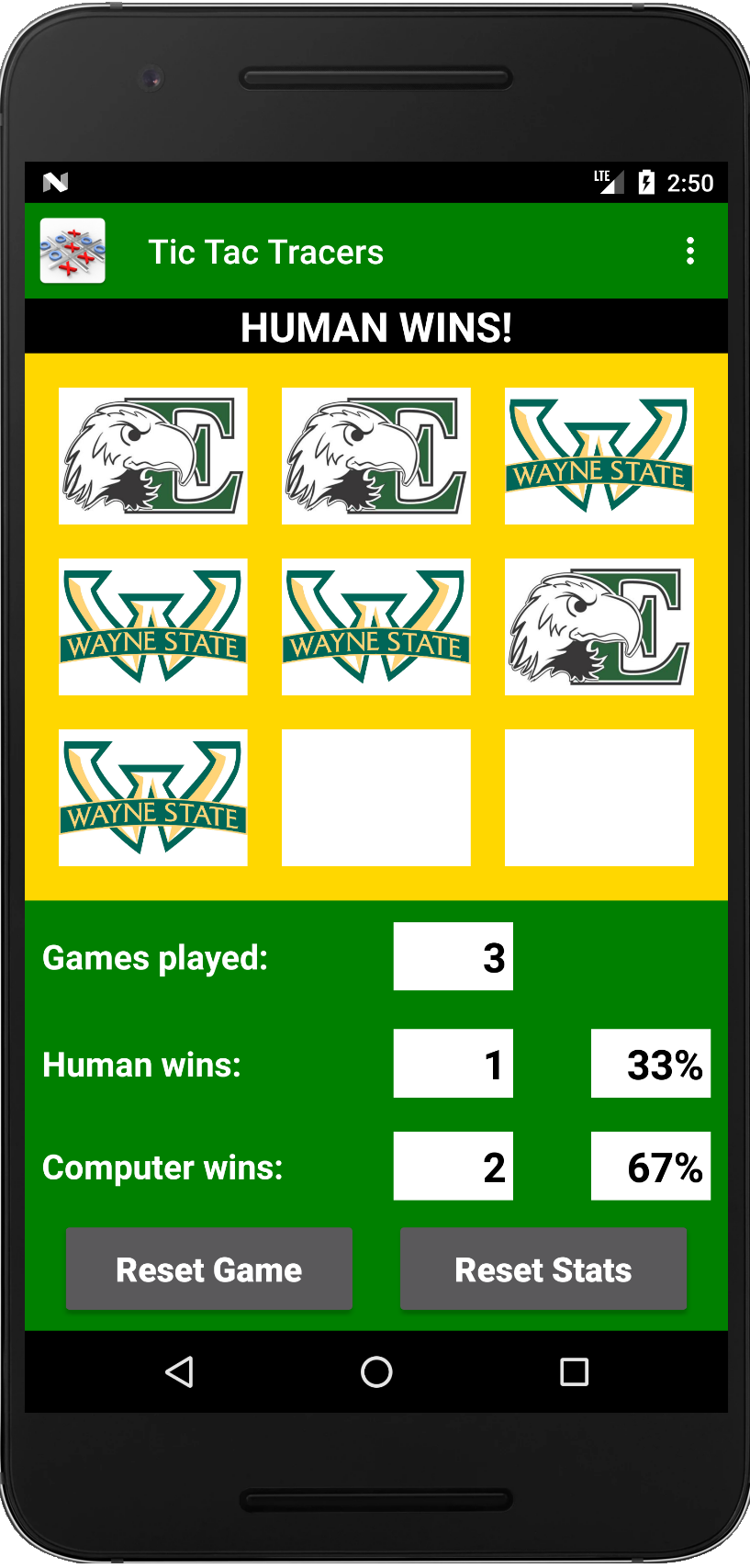
**Button section**

The button section has the following two buttons:

● Button **Reset board** resets the board so that all tiles are set to their background color. It first shows a yes-no dialog box to confirm that the user wants to reset the board.

● Button **Reset stats** resets the number of games played, human wins and percentage, and computer wins and percentage, and increments the global reset-stats count. It first shows a yes-no dialog box to confirm that the user wants to reset the wins.

Here is a sample Main screen:



**Main screen layout hints:**

● To size the tiles proportionally, use tag **Space** with a small width between tiles, and tag **ImageButton** with a layout weight of 1.

● To size the other controls proportionally, do the opposite: use tag **Space** with a layout weight of 1 between controls, and fixed sizes for the controls.

● To make a text box read-only, use attribute **android:focusable="false"**.

**Main screen logic hints:**

● Use an array to hold the nine image buttons. This makes it easier to apply logic to all of them in a loop.

● Use an array to hold the status (not selected, selected by human, selected by computer) of the nine image buttons. This makes it easier to determine if anyone has won. There are eight possible conditions for a player to win.

● The following functions are useful on image buttons:

setBackgroundColor(Color.parseColor(<hex-color>)); to apply a background color.

setImageResource(<image-resource-ID>); to apply an image.

setImageResource(0); to remove an image.

setEnabled(false); to prevent user from selecting it again.

setEnabled(true); to reset so user may select it.

● Use a simple algorithm to determine the next computer move. This can be randomly selecting the next open tile.

● When the app transitions from the Settings screen to the Main screen, apply any changed color settings.

**Settings screen**

The application has a **Settings screen** containing the following controls:

**Color settings section**

The stats section enables the user to select the board and tile colors. It has the following two rows of controls:

● A label, seek bar, and image for choosing the board background color. This may be a decimal range that is converted to a hex value. The image previews the color.

● A label, seek bar, and image for choosing the tile background color. This may be a decimal range that is converted to a hex value. The image previews the color.

**Image settings section**

The image section enables the user to select the pair of images that will represent the human and computer on the board. It has the following three rows of controls:

● A radio button, human image, and computer image. Select this by default.

● A radio button, human image, and computer image.

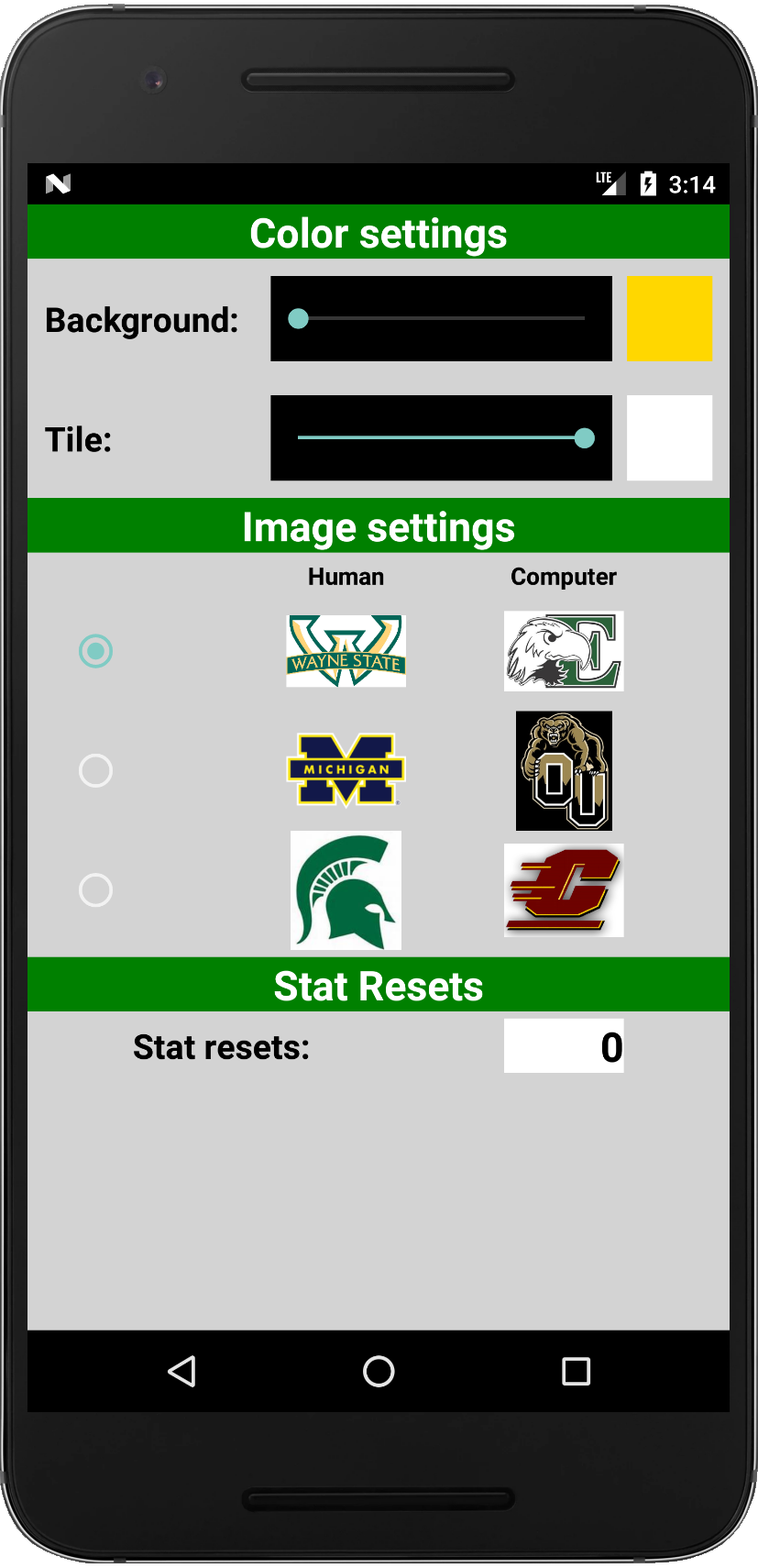
● A radio button, human image, and computer image.

**Stat Resets section**

The Stat Resets section enables the user to see the number of stat resets. It has the following row of controls:

● A label and read-only text box showing the overall number of stat resets.

Here is a sample Settings screen:



**Settings screen layout hints:**

● To size the controls proportionally, use tag **Space** with a layout weight of 1 between controls, and fixed sizes for the controls.

● For the image settings, place the radio buttons in a vertical layout, human images in a vertical layout, and computer images in a vertical layout. Use tag **Space** with a small width between vertical layouts.

● To select a radio button by default, use attribute **android:checked="true";**.

● To properly scale the displayed images, use attribute **android:scaleType="centerInside";**.

**Settings screen logic hints:**

● Convert the seek bar value from integer to hex with a leading "#".

● The following functions are useful on images:

setBackgroundColor(Color.parseColor(<hex-color>)); to apply a background color.

**Shared values**

The application needs to share data between the **Main** and **Settings** screens. Thefollowing values are useful:

● Board background color (string) contains a hex color value.

● Tile color (string) contains a hex color value.

● Image set (integer) is the selected image set.

● Human image (integer) contains a resource ID of the image.

● Computer image (integer) contains a resource ID of the image.

● Stat resets (integer) is the stat reset count.

Run the application and get a screenshot sequence of two complete games including changing the settings before each game.

**Main activity code**

*[your ActMain code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Main activity layout**

*[your laymain XML here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Main menu layout**

*[your mnumain XML here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Settings activity code**

*[your ActSettings code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Settings activity layout**

*[your laysettings XML here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Shared enum code**

*[your Shared enum code here]\**

**If possible, format your code like this:**

**Font “Courier New”**

**Font size “9”**

**Bold**

**Screenshots**

*[your Main and Settings activity screenshots for two complete games here]\*\**

**\* Copying-and-pasting code to a Word document**

**macOS**

1) From the Android Studio editor window, press **command-A** and press **command-C**.

2) From within the Word document, press **command-V**.

**Windows**

1) From the Android Studio editor window, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

**\*\* Copying-and-pasting a screenshot to a Word document**

**macOS**

1) From the Android Studio emulator or droidAtScreen utility, press **shift-command-4-space**.

2) From within the Word document, **command-V**.

**Windows**

1) From the Android Studio emulator or droidAtScreen utility, press **ALT-PrintScreen**.

2) From within the Word document, press **CTRL-V**.