Section 1: Description

This program turns on specific LEDs when certain buttons are pressed in a specific manner. Lab1 uses both push buttons on the launch pad as well as the Booster Pack to manipulate the red LED on the launchpad and both RGB LEDs on the Launch Pad and Booster Pack. By default, when none of the buttons are pressed, there should be no LEDs on. The following functionality had to be added in Lab1(**taken directly from Lab1 project statement**):

● Pushing and holding down the right button on the Launchpad turns the red and blue LEDs on in the RGB LED on the Launchpad. When on together they should produce what appears to be a bright purple color. As always when the button is released, the LEDs should return to being off.

● Pushing and holding down the top button (S1) on the BoosterXL turns the red, blue, and green LEDs on in the RGB LED on the BoosterXL. When on together they should produce what appears to be a bright white color. As always when the button is released, the LEDs should return to being off.

● Pushing and holding down the bottom button (S2) on the BoosterXL turns the red, blue, and green LEDs on in the RGB LED on both the Launchpad and BoosterXL. In total, you should see two bright white lights. As always when the button is released, the LEDs should return to being off.

● Pushing and holding down the left and right buttons on the Launchpad simultaneously turns on the red LED on the left side of the Launchpad, the blue LED on the right side of the Launchpad, and the green LED on the BoosterXL. In total, you should see three lights, from left to right they should be red blue green. As always when the button is released, the LEDs should return to being off.

● Pushing and holding down the top (S1) and bottom (S2) buttons on the BoosterXL simultaneously turns on both of the red LEDs on the Launchpad the red LED on the BoosterXL. In total, you should see three red lights. As always when the button is released, the LEDs should return to being off.

● Pushing and holding down the left and right buttons on the Launchpad and the top (S1) and bottom (S2) buttons on the BoosterXL simultaneously turns on the red LED on the left side of the Launchpad, the red blue and green LEDs on the right side of the Launchpad (which will appear white), and the blue LED on the BoosterXL. In total, you should see three lights, from left to right they should be red white and blue. As always when the button is released, the LEDs should return to being off.

In my program, I am able to implement turning on the left led with the left button of the launch pad being pressed, pressing the right button turns on a purple LED on launch pad, the top button on the booster pack turns on a single white LED on the booster pack, pressing the bottom button on the booster pack turns on the white LED on both the launch pad and booster pack, pressing both the right and left side buttons on the launch pad turns on red, blue, and green buttons on the launch pad and booster pack, and pressing the top and bottom button on the booster pack turns on all three red LEDs. However, when turning on all three LEDs I realized I

must press the top button then the bottom button at the same time to turn on all red LEDs. If I press them at the same time but I'm a split second off, different colored LEDs might be shown. I was unable to get all four buttons to display red, white, and blue because my booster pack LED displays a white LED opposed to the blue that needs to be displayed.

Section 2: Using #define to write portable code

I used #define in my code to define my constants specific to a bit that triggers a button and or LED, which could be found in the user manuals. Portable C code is code that complies with various compliers and runs the same on different platforms without modification. It is important to use #define when writing portable C code because this tells the preprocessor that a certain variable has a specific constant before passing it on to the compiler. This is important when running code on different machines with various compliers. By predefining a variable to a constant before compiling, we are able to reduce errors that may occur during the compiling process if that C complier has a different macro for said variable.