Name: ________________________

Constantly Variable

Professor Bob Brown – Kennesaw State University
Bob.Brown@Kennesaw.edu

Variables, Constants, and Data Types

In computing, a *variable* is a named area of memory that holds a value. So, variables have a *name* and a *value*. The name can usually be whatever the programmer wants. Variables can change value as a program runs.

A *constant* has a value, but usually not a name. The value is established by the programmer and does not change as a program runs.

Everything in a computer's memory is just ones and zeros. The bits 01000001 can represent the number 65, the capital A, or part of something else. To help programmers keep track of what the bits in a given area of memory mean, programming languages introduce the concept of *data type*, or just *type*. The MakeCode language for programming the Micro:Bit defines five data types. All MakeCode variables start out as numbers; it takes another step to change them to a different type. Once the type of a variable has been established, the variable can hold data of only that type.

- **Number:** A variable that is a number can hold either an *integer* (counting number) or a number with a decimal point, called a *floating-point number*. Examples are 1234 and 3.14159
- **String:** A string variable can hold letters, numbers, and punctuation. Examples are "Hello, World!" and "2190 Wallingford Drive." The length of a string variable can change as a program runs.
- **Boolean:** Boolean variables can hold only the values True and False. Boolean variables are named for George Boole, an 18th Century British mathematician and philosopher who did work on the mathematics of True and False. They're used for answering questions like, "Have we done this ten times yet?"
- **Array:** In MakeCode, an array can hold a list of items. Arrays may be different in other languages.
- **Sprite:** A sprite represents the position of a dot on the screen as two numbers, an X coordinate and a Y coordinate. We could have used two numbers for this, but having the *sprite* data type makes things like writing games easier.

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Last update: 2020-02-25

A Program to Keep Score

We're going to use this program for Rock-Paper-Scissors, but you could use it for any game of two players.

Algorithm

- Set scores to zero at start.
- When Button A is pressed
  - Add one to player A's score
  - Display an A in the LEDs
- When Button B is pressed
  - Add one to player B's score
  - Display an B in the LEDs
- When both are pressed, display the scores.

If the players tie, we can press both buttons, or press neither button.

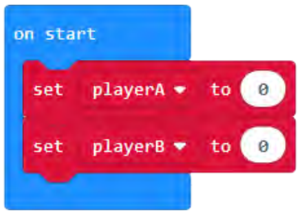
Start a New Program

Click the purple "New Project" box. Name your program *keepscore*. Keep the "on start: block on the screen, and drag anything else to the left into the trash can.

Make Number Variables and Set their Values

Click the *Variables* area in the center, then *Make a Variable*. It will be a *Number* variable when you make it. Name your variable *playerA*. Make another variable for *playerB*.

From the Variable area, drag a Set block for each variable into the "on start" block; that will set your variables to zero. Zero is a *constant*. You could change it to any other number, but for this program, we want to start at zero.

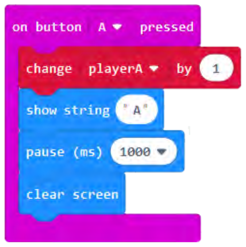


```
on start
    set playerA ▾ to 0
    set playerB ▾ to 0
```

Make Blocks for the Buttons

Make a block for "on button A pressed" that does the following:

- Change the *playerA* variable by 1
- Display "A" on the LEDs
- Pause for one second
- Clear the screen.



```
on button A ▾ pressed
    change playerA ▾ by 1
    show string "A"
    pause (ms) 1000 ▾
    clear screen
```

Make another block for button B. *Hint:* You can right-click in the purple area and choose "duplicate." Then you must change A to B in several places in the new block.

Finally, make a block to display the scores if both buttons are pressed. We'll talk about that in class.

Play a few rounds of Rock-Paper-Scissors and use your program to keep score!