**Team 8-Kumar-Wagh-Oboh**

**Team Members 1.** [Naveen Kumar](mailto:nk3090@nyu.edu)

**2.** [Aditya Wagh](mailto:amw9425@nyu.edu)

**3.** [Gordon Oboh](mailto:goo2006@nyu.edu)

**Project 1**

**Problem 1**

In problem 1, we are supposed to take input from user Num1 and Num2 and store and display the result as a difference of Num1 and Num2 whether it’s negative or positive on the terminal.

| '{$STAMP BS2} '{$PBASIC 2.5}  'Initializing variables to store inputs & results Num1 VAR Byte Num2 VAR Byte Result VAR Byte  DO  ' Ask the user to input number 1  DEBUG CR, "Enter Num1: "  DEBUGIN SDEC Num1   ' Ask the user to input number 2  DEBUG "Enter Num2: "  DEBUGIN SDEC Num2  ' Store the difference of x and y in Result  Result = Num1 - Num2   ' Check for the sign of the result if it's > 127, and do a bitwise negation.  IF (Result > 127) THEN  Result = ~Result+1  DEBUG "Result: "  DEBUG SDEC -Result   ' Print result if it's < 127  ELSE  DEBUG SDEC ? Result  ENDIF  LOOP |
| --- |

**Step1**: As you can see in the program above, we are declaring variables Num1, Num2 and Result first.

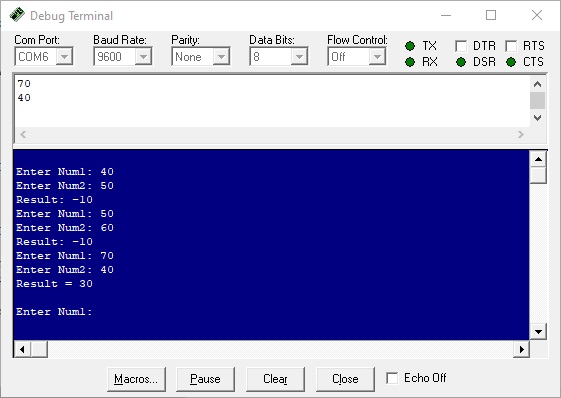
**Step2**: We are using the DEBUGIN command to allow the user to enter two variables Num1 and Num2

**Step3**: We are calculating the difference between Num1 and Num2 and storing it into the Result variable.

**Step4**: In the byte size variable, when the result is negative then we get the answer 256 - result. We have to represent the result in negative form. In order to do that we complement the result and add 1 in it if it’s bigger than 127 and get printed on the terminal. For example, if num1 = 30, and num2 = 40, the value of Result turns out to be 246. To find the actual decimal value of the absolute difference, we do a bit wise negation operation on Result, and add 1 to it.

Result = ~ Result + 1

If it’s less than 127 and greater than and equal to 0 then the result will be the same and get printed on the terminal.



The result is displayed on the terminal as you can see on the screen above.

**Problem 2**

This task requires us to create a basic Cartesian Coordinate system within the debug terminal.

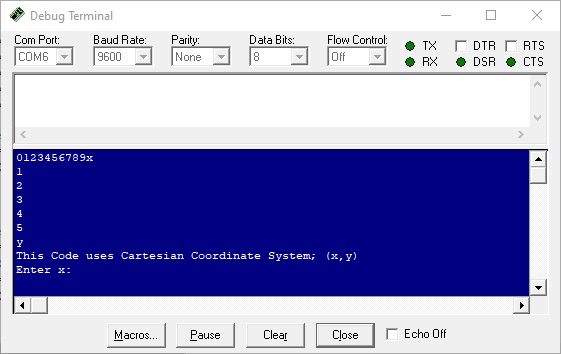
We used a FOR Loop to create the X-Axis and Y-Axis with 0-9 and 1-5 respectively,

The DEBUG command was used to display and give instructions to the user, the DEBUGIN command was implemented to accept two inputs from the user, one for the x- coordinate and the other for the y- coordinate, the program then placed an asterisk(\*) at the appropriate location with regards to the coordinate system.

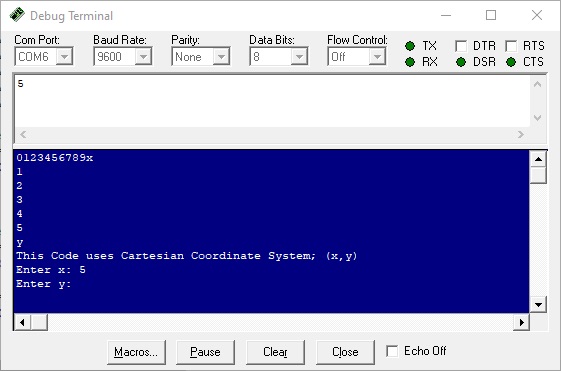
| ' {$STAMP BS2} ' {$PBASIC 2.5}  'Printing Coordinate axes reps VAR Nib 'Counting Variable x VAR Nib y VAR Nib i VAR Nib j VAR Nib  'X-Axis Representation FOR reps = 0 TO 9  DEBUG DEC reps NEXT DEBUG "x" 'To add X at the end  'Y-Axis Representation FOR reps = 1 TO 5  DEBUG CR, DEC reps NEXT DEBUG CR, "y" 'To add Y at the end  DEBUG CR, "This Code uses Cartesian Coordinate System; (x,y)" ' Ask the user to input x axis coordinate DEBUG CR, "Enter x: " DEBUGIN DEC x  ' Ask the user to input y axis coordinate DEBUG "Enter y: " DEBUGIN DEC y  DEBUG CLS  FOR reps = 0 TO 9  DEBUG DEC reps NEXT DEBUG "x", CR 'To add X at the end FOR i = 1 TO 5  DEBUG DEC i  FOR j = 1 TO 9  IF x = j AND y = i THEN  DEBUG "\*"  ENDIF  DEBUG " "  NEXT  DEBUG CR NEXT DEBUG "y" END |
| --- |

**Step 1**: In the Code, variables were declared and they were declared as **nibbles**  since we wouldn’t be counting above 15.

**Step 2**: A **FOR** Loop was used to create a number line for the X-Axis and Y-Axis, as shown in the figure below.



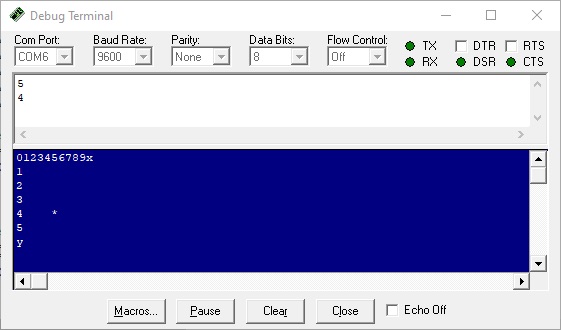
**Step 3**: **DEBUGIN** was implemented to accept two inputs from the user, first input and second input for the X-coordinate and Y-coordinate respectively, as shown in the figure below.



**Step 4**: Upon final user input **DEBUG CLS** clears the screen and the **FOR** Loop after that re-draws the 0 to x as shown in the figure below. Now there are two for loops one for x axis and another for y.

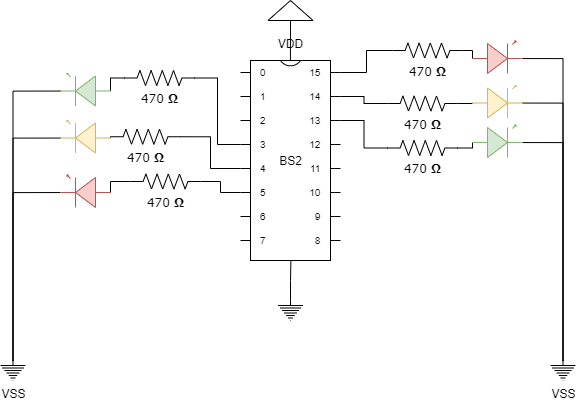
**Step 5**: The program traverses from left to right in row 1 and checks if the user input coordinate matches the current location or not mentioned in the If … else statement in the program above.

Similarly, the program traverses in another row and checks the coordinates.



**Problem 3**

In this task we are supposed to make a traffic light system for North-South and East-West directions. When vehicles are running in the North-South direction, then East-West traffic is stopped. When vehicles are running in East-West traffic, then the North-South traffic will be on stop. For North-South direction we are connecting Red LED with 15, Yellow LED with 14 and Green LED with 13 pin of Basic Stamp 2 microcontroller. For East-West direction we are connecting Red LED with 5, Yellow LED with 4 and Green LED with 3 pin of Basic Stamp 2 microcontroller as you can see in the circuit diagram.



**Calculations for Resistors**

The maximum source current per 8 I/O pins is 40mA.

We are connecting 3 LEDs at 3, 4 and 5 positions and 3 LEDs at 13, 14 and 15. So, the maximum current allowed per pin is 40/3 = 13.33mA. A LED will take 1.4 V and the resistor will take 3.6 V from 5V source. So, 3.6/13.33 = 270.06 Ohm. We need a resistor higher than that. So, we are connecting a 470 Ohm resistor.

**Program**

| ' {$STAMP BS2} ' {$PBASIC 2.5}  OUTPUT 3 'Green LED East-West OUTPUT 4 'Yellow LED East-West OUTPUT 5 'Red LED East-West OUTPUT 13 'Green LED North-South OUTPUT 14 'Yellow LED North-South OUTPUT 15 'Red LED North-South  DO  OUT15 = 1 'Red(North-South) LED ON  OUT3 = 1 'Green(East-West) LED ON  OUT5 = 0 'Red(East-West) LED OFF  OUT14 = 0 'Yellow(North-South) LED OFF  SLEEP 30  OUT15 = 1 'Red(North-South) LED ON  OUT4 =1 'Yellow(East-West) LED ON  OUT5 = 0 'Red(East-West) LED OFF  OUT3 = 0 'Green(East-West) LED OFF  SLEEP 10  OUT5 = 1 'Red(East-West) LED ON  OUT13 = 1 'Green(North-East) LED ON  OUT15 = 0 'Red(North-South) LED OFF  OUT4 = 0 'Yellow(East-West) LED OFF  SLEEP 30  OUT5 = 1 'Red(East-West) LED ON  OUT14 = 1 'Yellow(North-South) LED ON  OUT15 = 0 'Red(North-South) LED OFF  OUT13 = 0 'Green(North-East) LED OFF  SLEEP 10 LOOP |
| --- |

**Step 1**: In the program, first, we are declaring pins 3,4,5,14,14 and 15 pins as output pins.

**Step 2**: Now, we are declaring DO..LOOP for an indefinite time to run an infinite loop for traffic lights.

**Step 3**: For 30 seconds, we are providing 15 and 3 pin High output and 5 and 14 Low output which causes Red(North-South) and Green(East-West) light to turn on and Red(East-West) and Yellow(North-South) light to turn off.

Similarly for 10 seconds we are giving pin 15 and 4 high output and pin 5 and 3 low output which causes Red and Yellow light to turn on and Red and green light to turn off as mentioned in the comments in the program and further a 30 seconds and a 10 second cycle runs.

You can see the working of the program in the following video.

<https://drive.google.com/file/d/1eO6DLwH9J37Sgirt1K3frwmgLJFzRNFW/view?usp=sharing>

**Problem 4**

The task requires us to design a prototype for a pupil counting system using pressure pads at the entry and exit of a school.

We realized this using two push buttons, which mimic the pressure pads. We declared two input pins at PIN 0 and PIN 15. Following that, we declared “**student\_count**” to store the value of the number of students in the school at a point in time. We also need to store the current state of each button to reset the button after the button is pressed, and released. The variable “**button\_state\_1**” and the variable “**button\_state\_2**” represent the current state of **button 1** and **button 2.**

Our algorithm is as follows.

| if **IN0 & button\_state\_1 = 0,** then  student\_count = **student\_count + 1**  **button\_state\_1 = 1**  Print output to the screen  if **(student\_count > 0)** then  if **IN15 = 0 & button\_state\_2 = 0** then  student\_count = **student\_count - 1**  **button\_state\_2 = 1**  Print output to the screen |
| --- |

**DRAWBACKS**

The method described has the following drawbacks

* It doesn’t take into account the fact that a student might jump out of the window, and the machine would not know about it. And the count would be a false representation of the number of pupils in the building.
* A student could backpedal, which will cause the microcontroller to count multiple times. This method doesn’t take into account such issues.

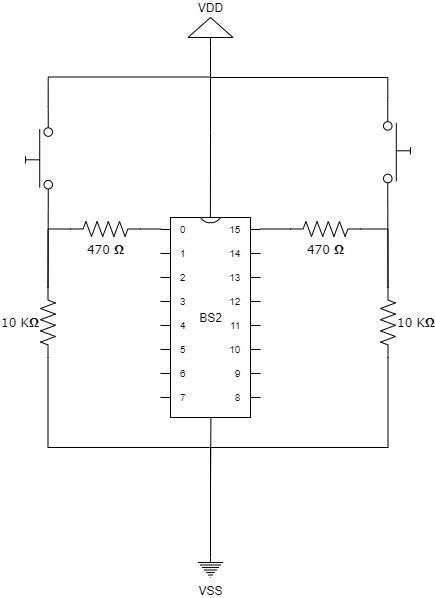
**IMPROVEMENTS**

The following methods can be applied to improved it;

* A Turnstyle could be used in place of a pressure pad, this would help to stop backpedalling.

| ' {$STAMP BS2} ' {$PBASIC 2.5}  ' Defining input pins for button 1 and button 2 respectively INPUT 0 INPUT 15  ' Defining variables to store the number of students in the premises ' and current state of each button student\_count VAR Word button\_state\_1 VAR Bit button\_state\_2 VAR Bit  student\_count = 0 ' Total number of students in the building button\_state\_1 = 0 ' Initial state of button 1 button\_state\_2 = 0 ' Initial state of button 2  Main:   ' Increase the student count on entry - mimicked by a button press  IF IN0 = 0 AND button\_state\_1 = 0 THEN  student\_count = student\_count + 1  button\_state\_1 = 1  DEBUG "Number of students in the school = ", DEC student\_count, CR  ENDIF   ' Reset state of button 1 to zero  IF IN0 = 1 THEN  button\_state\_1 = 0  ENDIF   ' Check if student count is zero, since it can't be negative  IF (student\_count > 0) THEN   ' Decrease the student count on exit - mimicked by a button press  IF IN15 = 0 AND button\_state\_2 = 0 THEN  student\_count = student\_count - 1  button\_state\_2 = 1  DEBUG "Number of students in the school = ", DEC student\_count, CR  ENDIF   ENDIF   ' Reset state of button 2 to zero  IF IN15 = 1 THEN  button\_state\_2 = 0  ENDIF  GOTO Main |
| --- |

**Schematic for the problem**



Circuit Diagram for 1.4

A link to the video of this Project has been provided here - [Project 1.4.MOV](https://drive.google.com/file/d/1XuH3MRIu5hAoNlp5KCP-B_2oY_C9CsN5/view?usp=sharing)