ثانويـة التكنولوجيا التطبيقيـة
Applied Technology High School

Robotics II

Module 2: Application of Data Programming Blocks

PREPARED BY

Academic Services Unit

December 2011

Module 2: Application of Data Programming Blocks

Module Objectives

Upon successful completion of this module, students should be able to:

- Understand the use of data programming blocks and their applications
- Understand the basic logic operations and apply them in programming robots.
- Perform mathematical operations using the math block.
- Control switches/loops using range and compare blocks
- Create variables and constants.
- Use the random block to personalize your robot

Module Contents:

1.1 Introduction

As you studied in module 1, data programming blocks can be found in the complete palette. They are designed to handle data in a variety of useful ways. For example, you can use the logic block to combine multiple conditions, the math block for calculating the distance your robot has travelled and the variable block to create variables and update them such as the final score in a pinball game or the total products produced in a factory. Figure 2.1 shows the various data blocks.

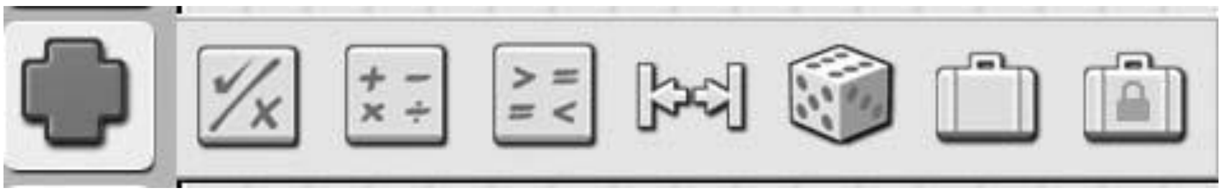


Figure 2.1: Data Blocks

In this module, you will study each of the data blocks, their configurations and applications.

1.2 Data Programming Blocks and their Applications

- **Math Block:** The Math Block is used to perform a mathematical operation on two input integer numbers. The result of the operation is available on an output data plug. The math operations are addition, subtraction, multiplication, and division. In the NXT-G 2.0 version, you can also calculate the absolute value and square root. Figure 2.2 shows how the math block looks when you add it to your program.

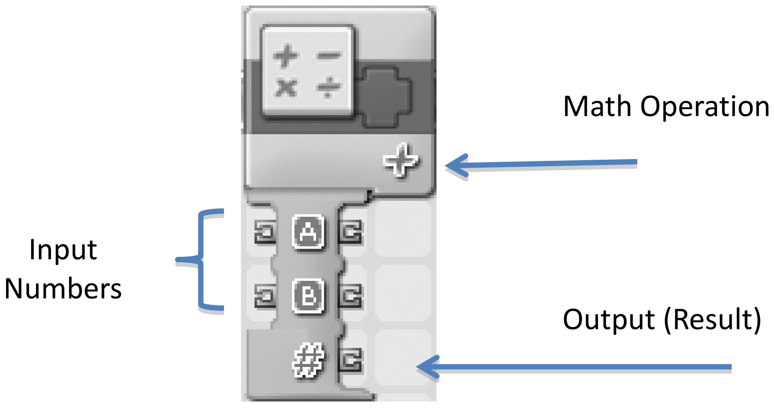
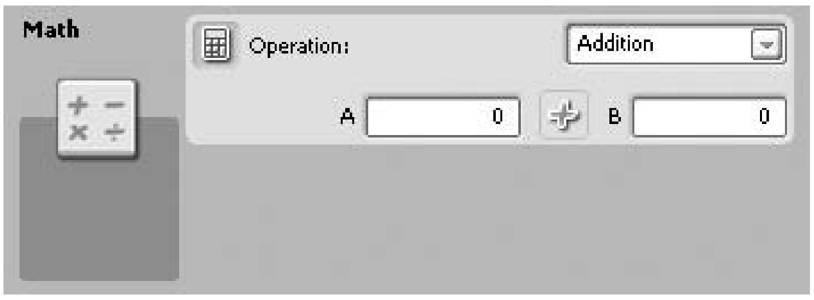


Figure 2.2: The Math Block

In the math block, A and B values can either be manually set in the configuration panel (shown in Figure 2.3) or dynamically set with input data wires.


Math

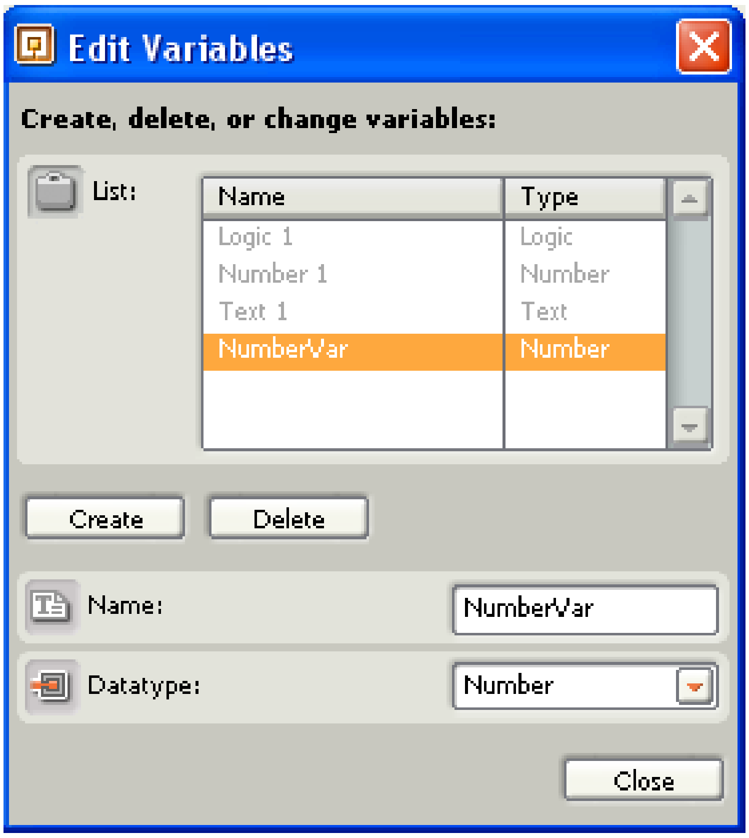
Operation: Addition

A 0 B 0

Figure 2.3: Configuration Panel for the Math Block

Using the math block, you can calculate the area of an object, the total distance travelled by the robot or modify a sensor reading to reflect the score in a pinball game.

- **Variable Block:** The variable block stores and retrieves variable values. You can save sensor readings in a variable and then update them after performing mathematical operations. For Example, you can count the total number of red or blue objects produced in a factory.
  Using variables in NXT-G is a two-step process. First you create the variable, and then you use the Variable block in your program to work with the data contained in the variable.
  To create a variable, go to edit → define variables


Edit Variables

Create, delete, or change variables:

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
NumberVar	Number

Name: NumberVar

Datatype: Number

Figure 2.4: Define Variables Window

As you can see in Figure 2.4, we created a new variable entitled "**NumberVar**" with the "**Number**" Datatype. You can create other data types such as "**logic**" or "**text**". Next, we can configure a Variable block to either read or write to the NumberVar variable that we created. Figure 2.5 shows the variable block.

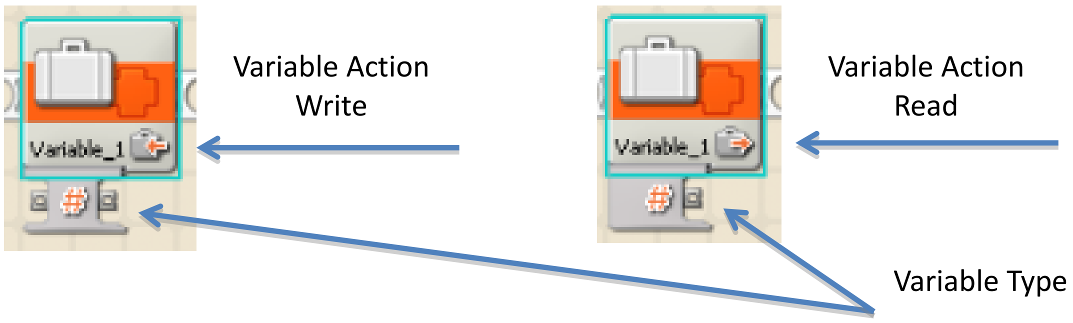
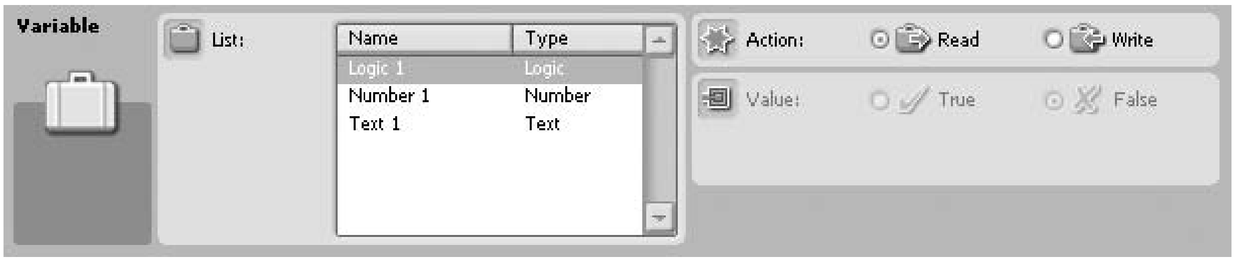


Figure 2.5 The Variable Block

When working with a variable block, you need to tell NXT-G which variable to use and whether to *read to* or *write from* the variable. To do so, use the Configuration Panel (Figure 2.6) to select the variable from the list, and then set the Action item to either Read or Write. When <u>Read</u> is selected, the <u>variable's current value is put on the data wire</u> attached to the output data plug. When <u>Write</u> is selected, <u>a value is stored in the variable</u>. That value can be supplied either using the Configuration Panel or via an input data wire. Usually, you will need to use the Configuration Panel to set a variable's initial value at the beginning of a program, and then use a data wire if you want to change the value later in the program.


Variable

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

Action: Read Write

Value: True False

Figure 2.6: Configuration Panel for the Variable Block

- **Range Block:** This block can take up to three inputs. It determines whether a test number is <u>inside</u> or <u>outside</u> a range of numbers. It outputs a logic value depending on the result of the comparison. Figure 2.7 shows how the range block looks when you add it to your program.

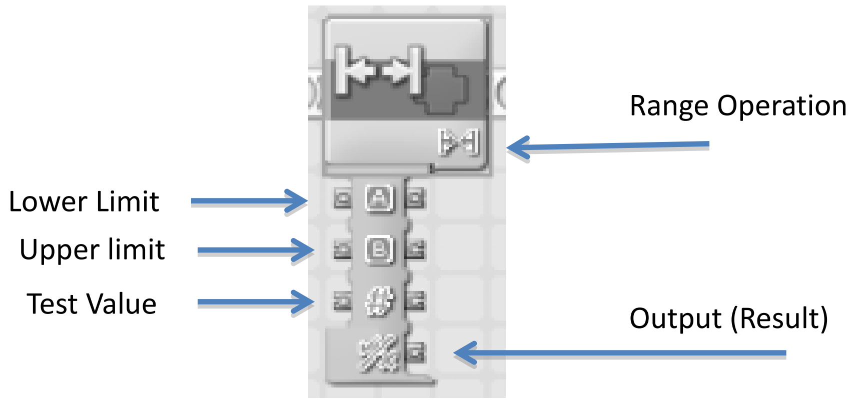
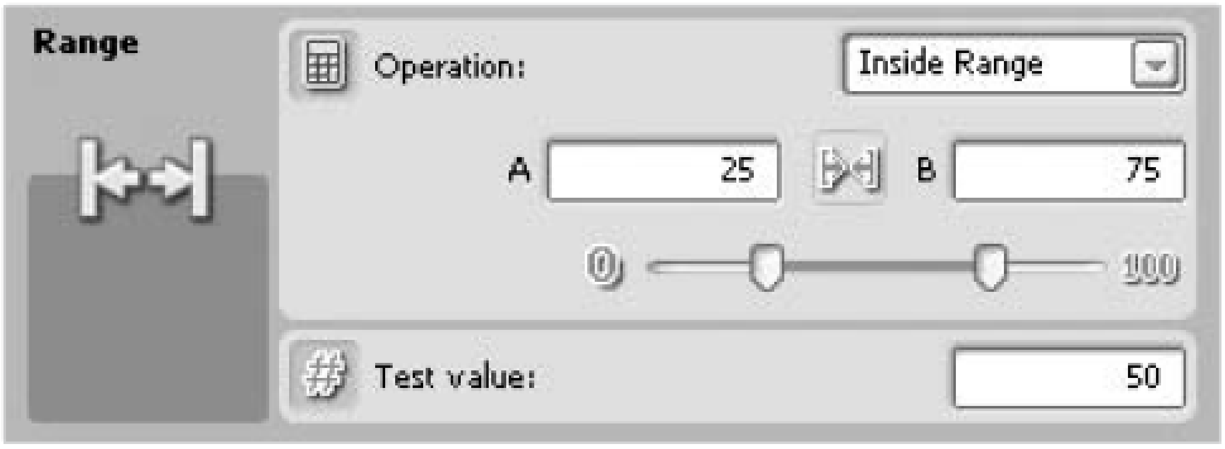


Figure 2.7: The Range Block

Figure 2.8 shows the Configuration Panel for the range block. You can set the range by using the two-sided slider, by entering the values in the A and B boxes or by supplying the two range limit values using data wires.

Using the slider, you can select lower and upper limits between 0 and 100. To use values greater than 100 or less than 0, enter the numbers in the boxes.


Range

Operation: Inside Range

A 25 B 75

Test value: 50

Figure 2.8: Configuration Panel for the Range Block

The test value can be entered into the box on the Configuration Panel or supplied by using a data wire.

Using the range block, you can improve on traditional color sorting codes by specifying a range values for each color.

Conduct Lab Activity 1 (Task 1 & Task 2) on page 15

- **Logic Block:** This block is designed to perform a logical operation on its inputs and output a true/false logical value. Its operations consist of And, Or, XOr, and Not. Figure 2.9 shows how the logic block looks when you add it to your program.

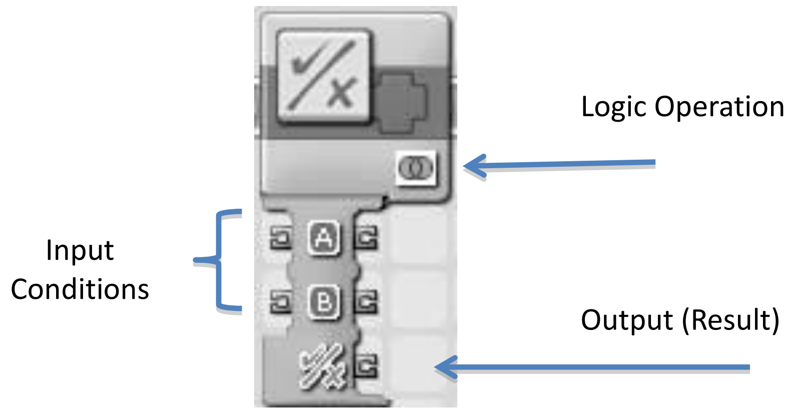


Figure 2.9: The Logic Block

To use the logic block, you select the operation you want to perform and supply the input values, using either data wires or the Configuration Panel (shown in Figure 2.10). Buttons are used to set the values in the Configuration Panel, with the (✓) meaning true and the (✗) meaning false.

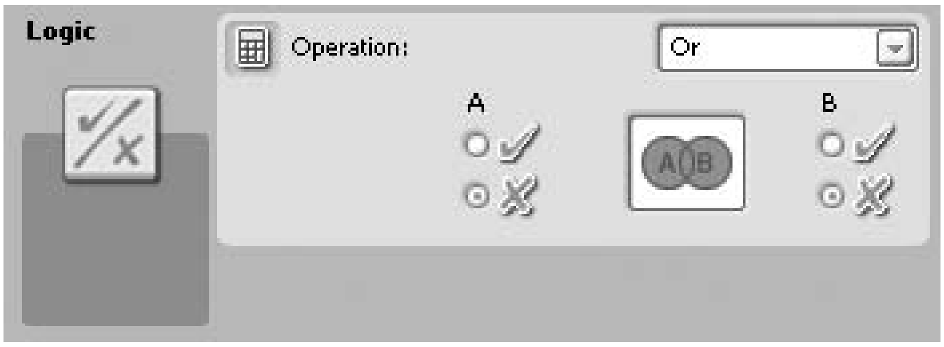
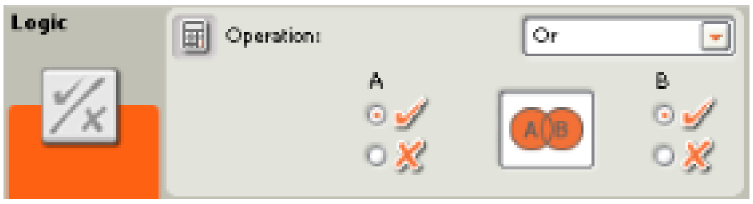


Figure 2.10: The Configuration Panel for the Logic Block

The Logic block supports four operations: And, Or, Xor, and Not. The key to using this block successfully is understanding what each operation does.

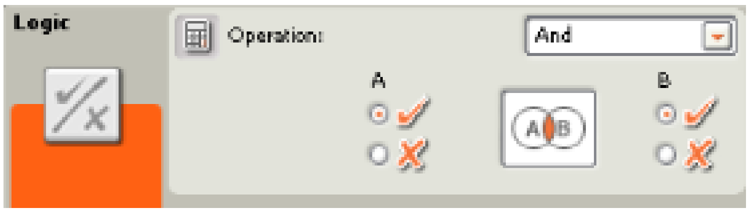
- **Or:** The result of the Or operation will be true if either input value is true or if both input values are true. The result will be false only if **both** input values are false.

Example: you want your robot to change direction if the ultrasonic value is less than 20 cm or the touch sensor is pressed.



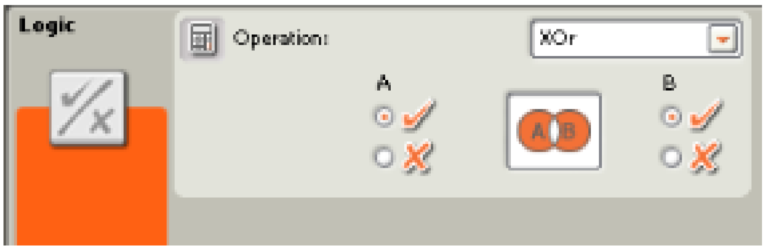
- **And:** The result of the And operation will be true only if both input values are true. If either input value is false, then the result will be false.

  Example: you want your car to operate when the ignition key is ON and all the doors are closed.

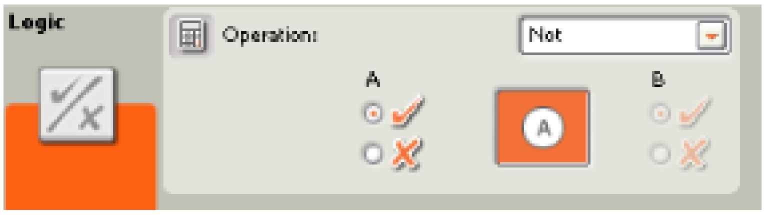


- **Xor:** Xor is an abbreviation for Exclusive Or. This is similar to the Or operation except that the result is false if both input values are true.

  Example: you want your robot arm to pick either candy or ice cream but not both.



- **Not:** This operation uses only the A input value and generates the opposite value. If the input value is true, then the output value will be false, and if the input value is false, then the output value will be true.



Logical operations are often described using a table that lists all the possible input values and the result for each operation. This is called a truth table (see Figure 2.11)

input A	input B	or	and	xor	not
False	False	False	False	False	True
False	True	True	False	True	True
True	False	True	False	True	False
True	True	True	True	False	False

Figure 2.11: Truth Table for the Logic Block

Conduct Lab Activity 1 (Task 3) on page 20

- **Constant Block:** Constant Block is used to access constants in your program. This block looks like the Variable block, with a lock added to show that the value cannot be changed. Figure 2.12 shows the constant block while Figure 2.13 shows the Configuration Panel for the constant block.

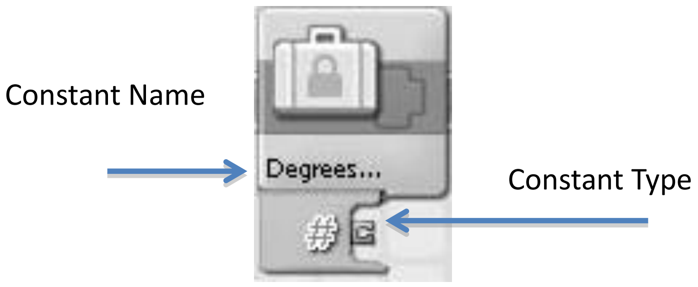
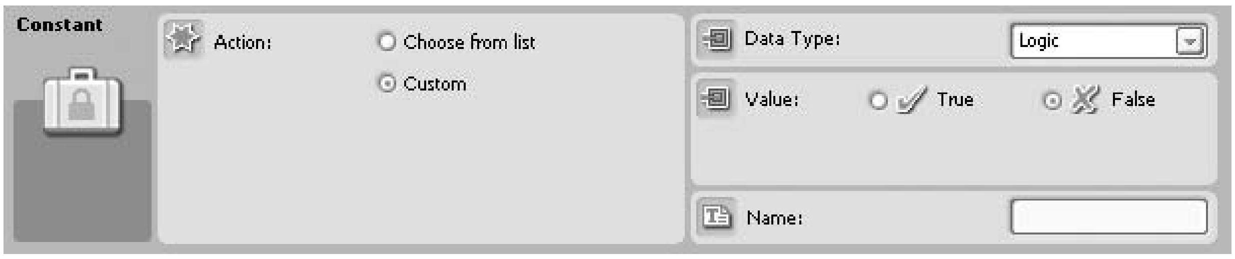


Figure 2.12: The Constant Block


Constant

Action: ( ) Choose from list [✓ Custom]

Data Type: Logic

Value: ( ) True [✓ False]

Name:

Figure 2.13: Configuration Panel for the Constant Block

The Action item determines how you will define the value to use, either selecting a constant from the list created using the Edit → define constants dialog or creating a custom constant.

The Data type can be logic, text or number. You can set the name and the value of the constant block by entering the values in the configuration panel. The name of the constant will be shown on the block.

To use the constant value, simply connect a data wire to the Constant block's output data plug.

- **Compare Block:** This block simply compares two numbers as to whether or not the first number is greater than, less than, or equal to the second number. It outputs a logic value depending on the result of the comparison. Figure 2.14 shows how the compare block looks when you add it to your program.

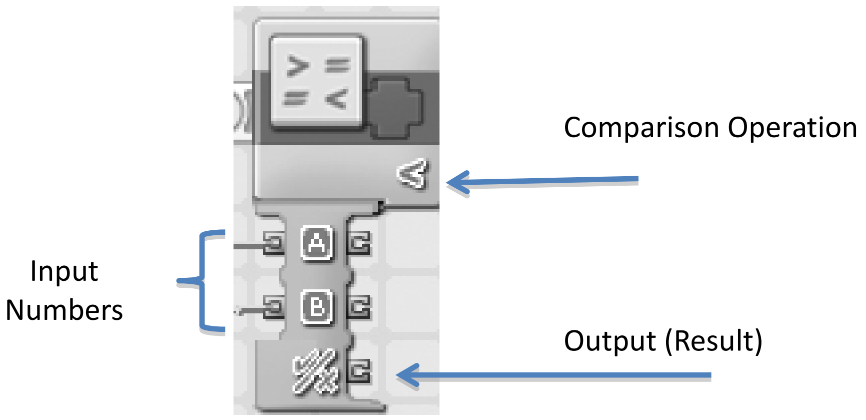
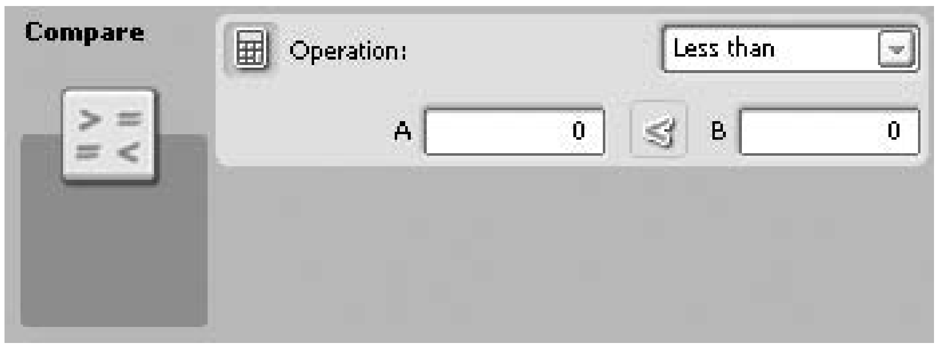


Figure 2.14: The Compare Block

You can supply the two input values using data wires or the Configuration Panel (shown in Figure 2.15).



Operation: Less than

A 0 B 0

Figure 2.15: Configuration Panel for the Compare Block

When the block runs (Figure 2.14), it takes the two input numbers, compares them, and puts the resulting logic value on the output data wire. For example, if the A value is 7 and the B value is 12, the result will be true, because 7 is less than 12. On the other hand, if A is 25 and B is 8, the result will be false, because 25 is not less than 8.

Using the compare block, you can compare the readings from two ultrasonic sensors. You can then use the result of the comparison to control a Switch or Loop block to determine which way your robot should go.

- **Random Block:** Random Block is used to generate a random number. It takes two inputs in the form of the min and max values of the random numbers it will be allowed to generate. It is used to create robotic games or a robot that exhibits unpredictable behavior because the block's output will vary each time the program runs. Often a robot that is a little unpredictable can be more interesting or seem to have more personality. Figure 2.16 shows how the random block looks when you add it to your program while Figure 2.17 shows the configuration panel for the random block.

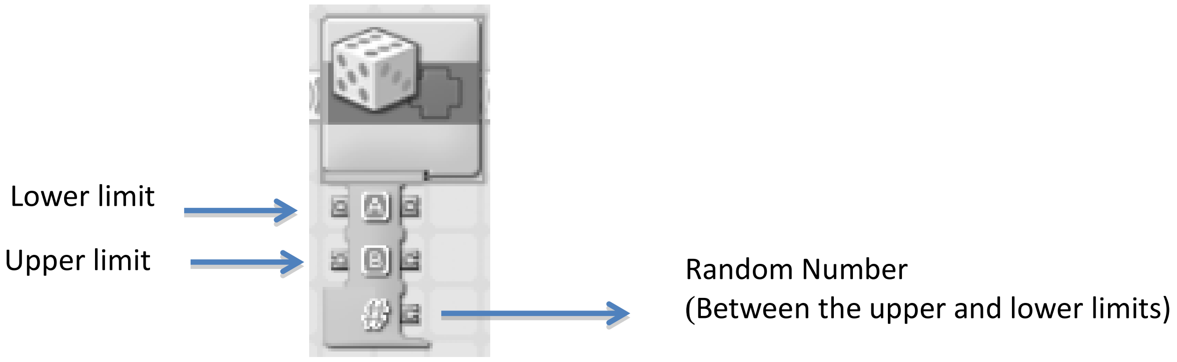
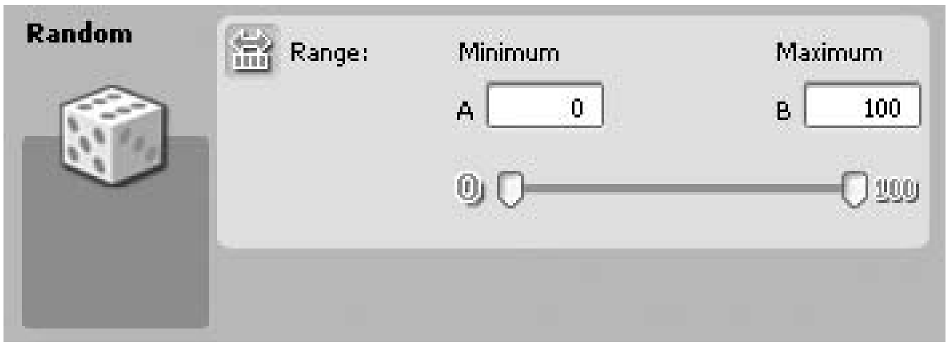


Figure 2.16: The Random Block



Random

Range: Minimum A 0 Maximum B 100

Figure 2.17: Configuration Panel for the Random Block

Like the range block, you can set the range by using the two-sided slider, by entering the values in the A and B boxes, or by supplying the two range limit values using data wires. Using the slider, you can select lower and upper limits between 0 and 100. To use values greater than 100 or less than 0, enter the numbers in the boxes. The minimum value for the lower limit is 0 while the maximum value for the upper limit is 32767.

Challenge: "Test your knowledge"

A steering value has a range between -100 and 100, how can you generate a random negative value to allow steering to take place, draw the configuration needed in the space provided.

Conduct Lab Activity 2 on page 22

1.3 Lab Activity 1

Objectives:

1. Understand the use of variable, math, range and logic blocks
2. Use the math block in counting applications to display the score of various games
3. View the final game score on NXT Screen

Material per Group:

4. 1 NXT Brick
5. 1 USB cable
6. 1 touch sensor
7. 1 ultrasonic sensor

Background Information:

All types of games involve the display of the final score. Every time you hit an object, you will have a certain score. The more you hit the object, the more score you will get at the end. Pinball and basketball machines are examples of games that show the final score. Figure 2.18 shows an example of pinball machine using the Lego NXT.

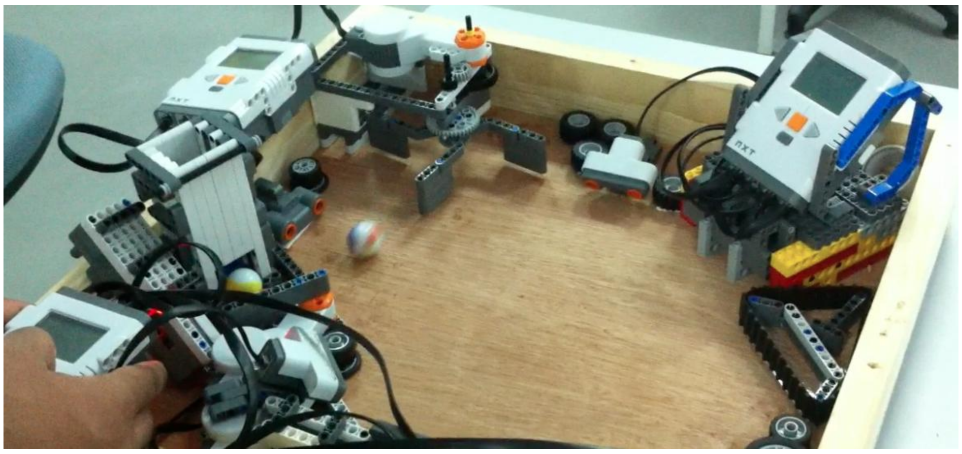


Figure 2.18: Pinball Machine using NXT Brick

In this lab activity, you will display the final score of a pinball machine on the NXT screen. Every time the touch sensor is bumped, the score will be incremented by ten. Furthermore, if the ultrasonic sensor reading is between 10 and 20 cm, the score will also be incremented by 10. The game will be over after two minutes.

Procedure:

1. Connect the touch sensor to port 1 of the NXT
2. Connect the ultrasonic sensor to port 4 of the NXT
3. Create a new program and name it **pinball**
4. Go to Edit → Define variables→ Create
5. Name the variable "Score" and select the Datatype "Number"

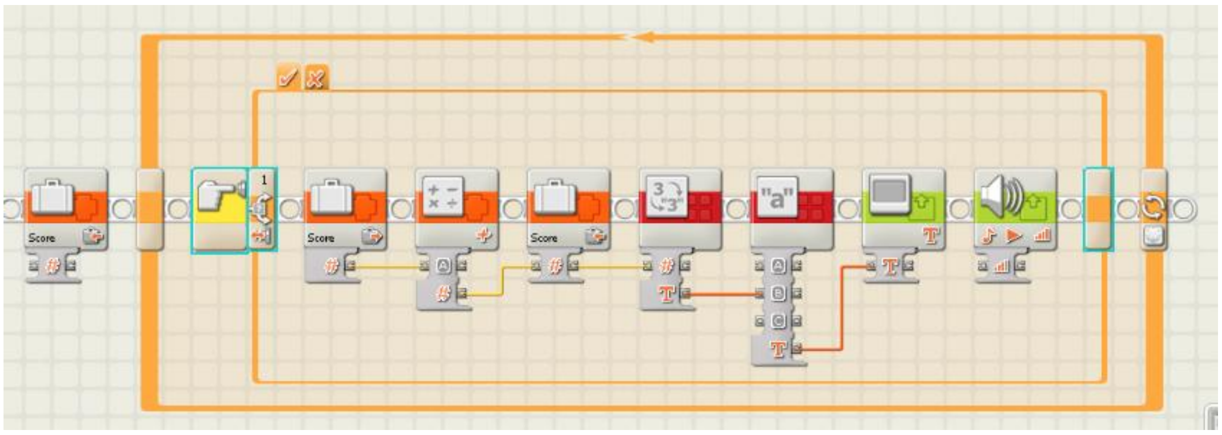
Programming Steps:

Task 1: Display the score of the touch sensor till the game is over

To display the score of the touch sensor, the following steps should be followed:

Steps	Programming Block/s
Determine if the touch sensor is bumped	Switch controlled by touch sensor
Define a variable to save the score Initialize the score to zero	Variable Block
If the touch sensor is bumped, the score will be incremented by 10	Math Block
The score value will be displayed on NXT	Text, Number to text and display blocks
Repeat updating the score till the game is over (time = 2 min.)	Loop Block

6. Write the following NXT-G code:



7. Configure the programming blocks as follow:

Programming Block	Configuration
Variable	• List→Score • Action→write • Value→ 0
Loop	• Control→ time • Time → 120
Switch	• Control→ Sensor • Sensor→ touch • Action→ bumped • Port → 1 • Uncheck Flat View
Variable	• List→Score • Action→read
Math	• Operation→ addition • B→ 10
Variable	• List→Score • Action→write
Number to text	-
Text	• A→ Score=
Display	• Action → text • X→0 • Y→ 40
Sound	• Action→ tone • Note→ B for 0.25s

8. Download and run your program
9. Bump the touch sensor, what is the value displayed on the NXT Screen? ____________
10. Repeat step 9 for 5 times. Record your results in the table below:

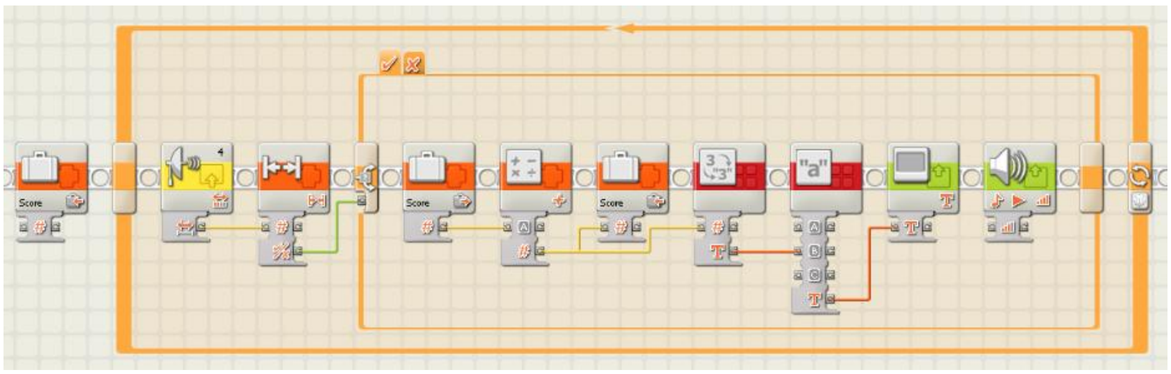
Number of times the touch sensor is bumped	Total Score Displayed on the NXT Screen
1	
2	
3	
4	
5	

Task 2: Display the score of the ultrasonic sensor till the game is over

To display the score of the ultrasonic sensor, the following steps should be followed:

Steps	Programming Block/s
Read the ultrasonic sensor value	Ultrasonic sensor Block
Determine if the ultrasonic sensor values is between 10 to 20 cm	Range and switch Block
Define a variable to save the score Initialize the score to zero	Variable Block
If the ultrasonic sensor values is between 10 and 20 cm, the score will be incremented by 100	Math Block
The score value will be displayed on NXT	Text, Number to text and display blocks
Repeat updating the score till the game is over (time = 2 min.)	Loop Block

11. Update the task 1 code to the following code:



12. Configure the newly added programming blocks as follow:

Programming Block	Configuration
Ultrasonic Sensor	• Port→4 • Show→ cm
Range	• Operation→ inside range • A→10 • B→20
Switch	• Control→ Value • Type→ logic • Uncheck Flat View

13. Download and run your program

14. Put your hand 30 cm away from the sensor, what is the value displayed on the NXT Screen? ____________

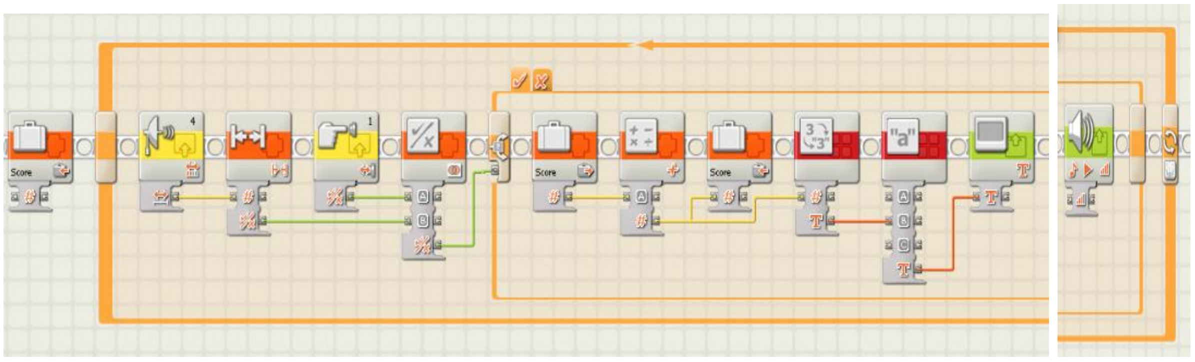
15. Repeat step 9 for the following distances. Record your results in the below table:

Distance (cm)	Total Score Displayed on the NXT Screen
30	
20	
15	
10	
5	

Task 3: Combing task 1 and task 2 in one code

The score of the pinball machine will be updated if the touch sensor is bumped **<u>or</u>** if the ultrasonic sensor range is between 10 and 20 cm. To combine both tasks, the <u>logic block</u> will be added to your code.

16. Update the task 2 code to the following code:



17. Configure the newly added programming blocks as follow:

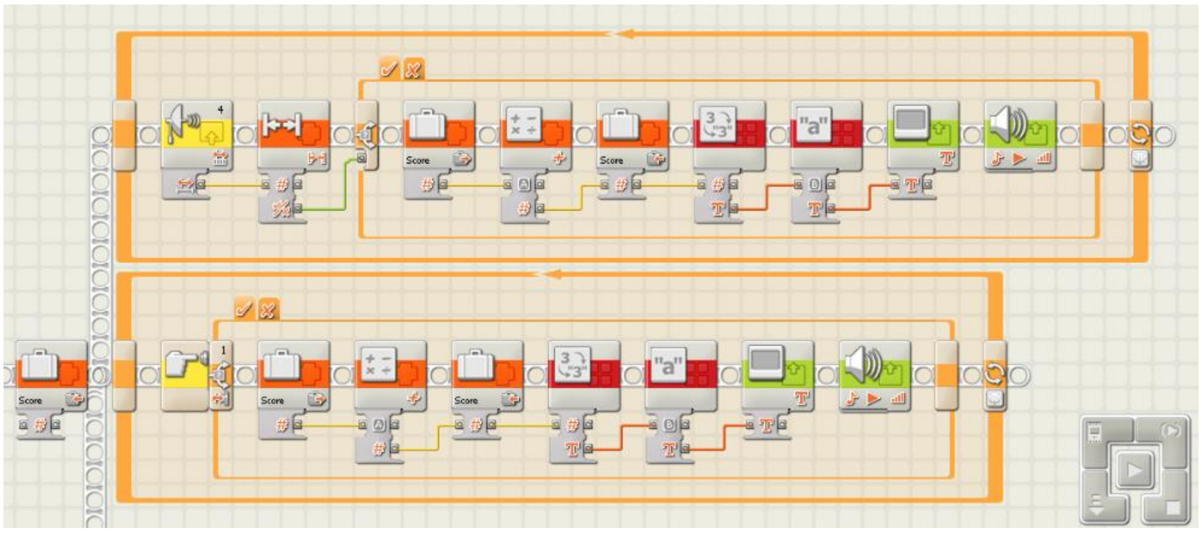
Programming Block	Configuration
Touch sensor Block	• Port→1 • Action→ bumped
Logic Block	• Operation→ OR

18. Put your hand 15 cm away from the sensor. At the same time, bump the touch sensor.

19. Download and run your program. What is the value displayed on the NXT Screen? _______

Challenge: The displayed value was 10 while the true value should be 20. How can you solve this problem? Modify your code to update the score twice if the touch senor is bumped and the ultrasonic sensor reading is between 10 and 20 cm.

Sample Solution:



1.4 Lab Activity 2

Objectives:

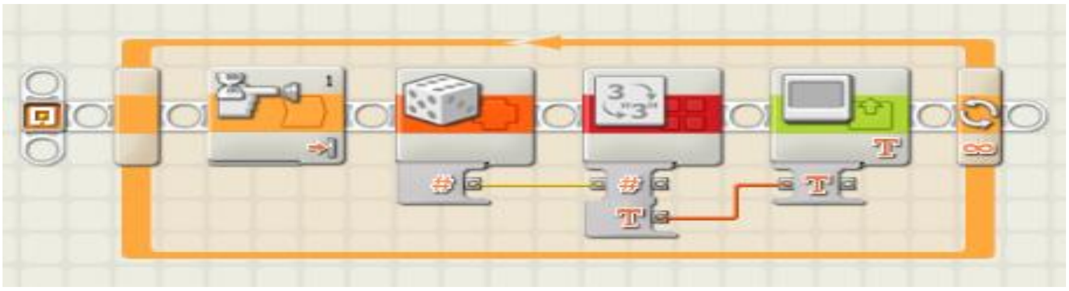
1. Understand the use of random block
2. Display the values of the random block on the NXT Screen
3. Control the motor steering randomly

Material per Group:

1. Explorer Robot (See Module 1)
2. 1 USB cable
3. 1 motor
4. 1 touch sensor

Procedure:

1. Connect the touch sensor to port 1 of the NXT
2. Create a new program and name it **randomtest**
3. Write the following NXT-G code:



4. Configure the programming blocks as follow:

Programming Block	Configuration
Loop	• Control→ Forever
Wait	• Control→ Sensor • Sensor→ touch • Action→ pressed • Port → 1
Random	• Minimum→ 0 • Maximum→ 200
Number to text	• -
Display	• Action → text

5. Download and run your program.
6. Press the touch sensor. What will be the value on the NXT Screen?

   ____

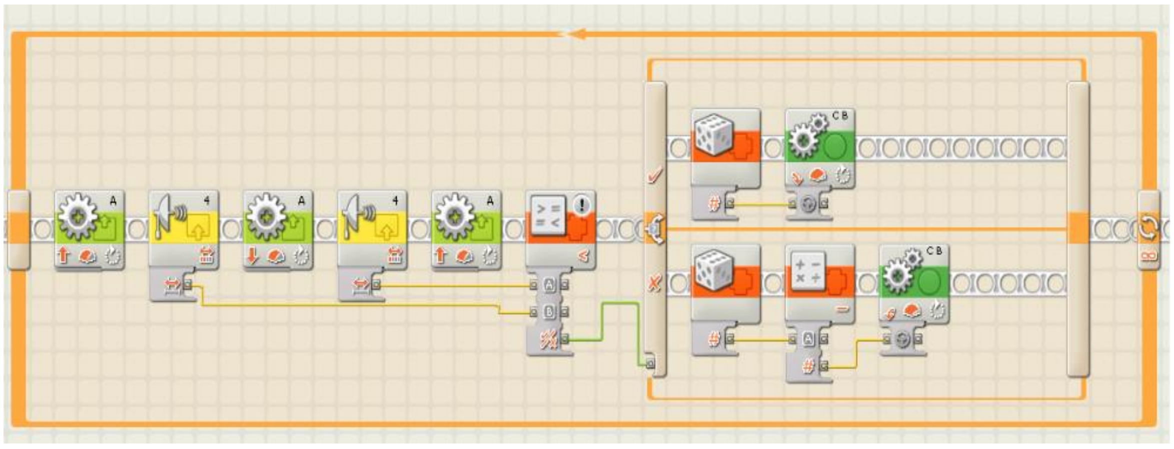
7. Repeat "step 6" five times. Record your values in the below table:

Number of trials	Number Displayed on NXT
1	
2	
3	
4	
5	

Questions:

- ➢ Are the displayed values the same every time? __________
- ➢ Is there is a value above 200? ________

8. Now, we will modify the code of the explorer robot (done in module 1) to steer the motors randomly.
9. Write the following NXT-G Code



10. Configure the newly added blocks as follow:

Programming Block	Configuration
Random Blocks	• Minimum→ 0 • Maximum→ 100
Math	• Operation → subtraction • B→ 100

11. Download and run your program several times.
12. Place an obstacle to the right side of the robot. Observe the direction of the turn and the amount of steering? ________________
13. Place an obstacle to the left side of the robot. Observe the direction of the turn and the amount of steering? ________________

Questions:

- What is the use of the math block in the modified explorer program?

  ____

  ____

- What is the effect of adding a random block to the steering control of the explorer?

  ____

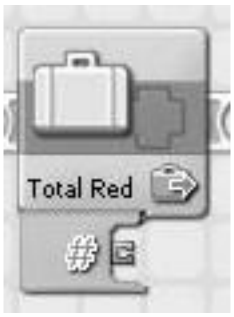
  ____

1.7 Review Exercise

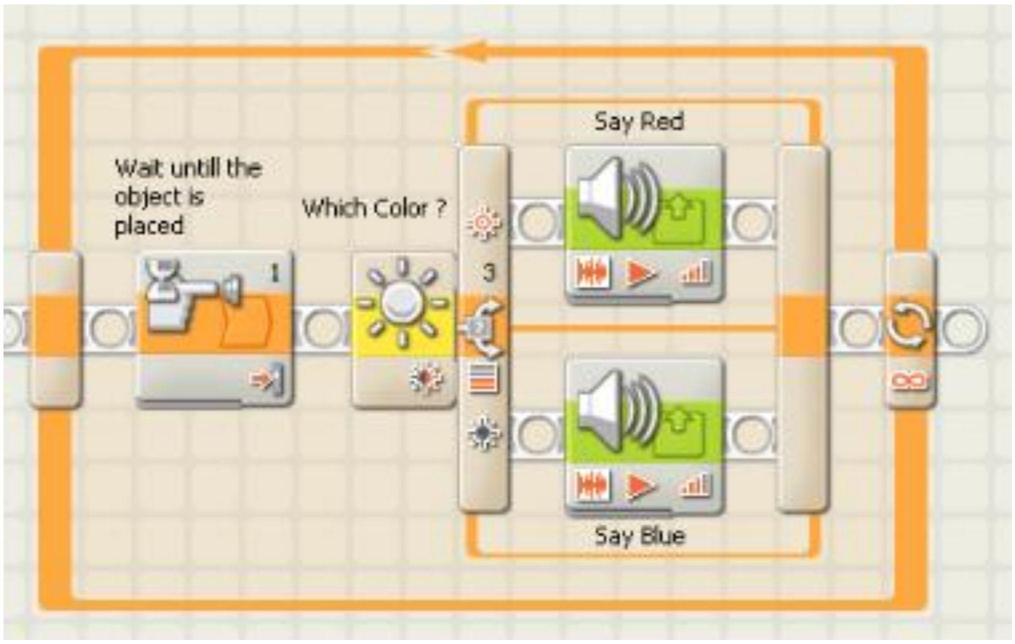
1. Compare between Logic, Math, Compare and Range blocks. Fill the required information in the table below:

	Logic	Math	Compare	Range
Block operation				
Type of inputs				
Type of output				
Number of inputs				

2. Mark (T) for true and (F) for false statements.

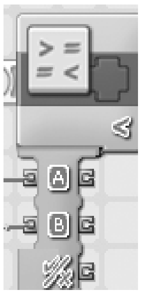
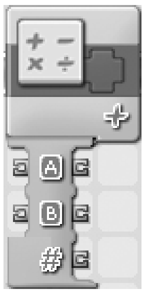
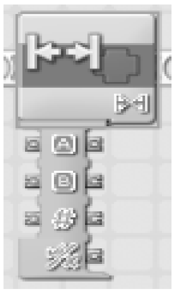
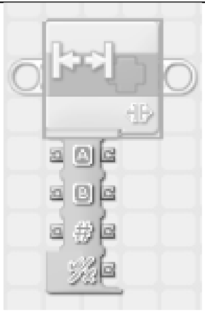
No.	Statement	T/F
1	The result of the Or operation will be true only if either input value is true.	
2	The Random Block can generate negative values.	
3	The action of the following variable block is "Read". 	
4	In all versions of NXT software, math block can perform the absolute value operation	

3. The following code is used to determine the color of an object (Red or Blue). Update the following program to count and display the number of red and blue objects. (Use the software to update the program)



Write your modified program here:

4. Determine the value of the output of the following programming blocks.

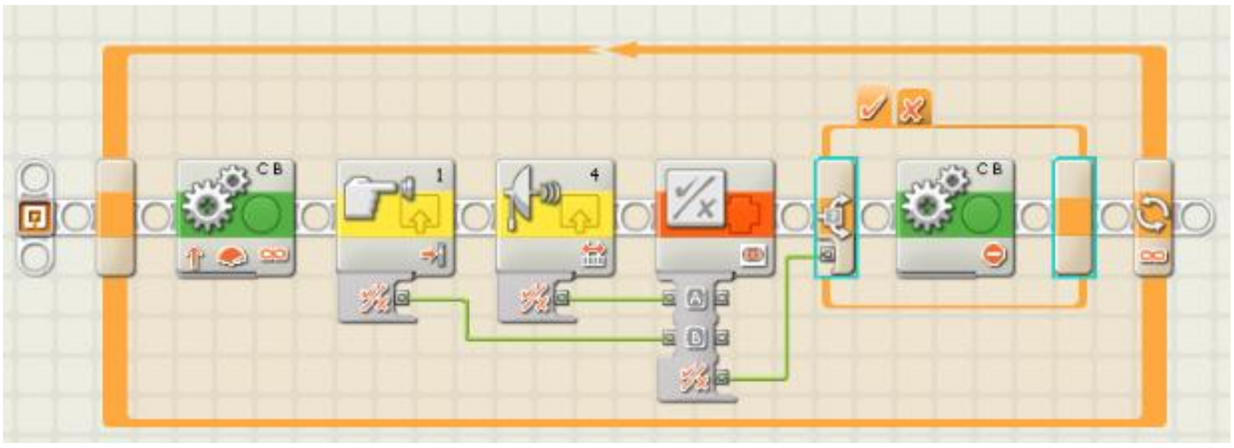
Block				

Conditions	A=10
	B=5
		A=20
		B=7
			A=5
			B=15
			#=6
				A=5
				B=15
				#=6

Output Value				

5. Explain the sequence of the following NXT-G Code:

(Note: the logic operation is Or, the Distance is < 25 cm)



1. ______________________________
2. ______________________________
3. ______________________________
4. ______________________________
5. ______________________________
6. ______________________________

References:

- Terry Griffin (2010). *The Art of LEGO MINDSTORMS NXT –G Programming*. No starch press: USA

- Kelly, F.G. (2007). *LEGO MINDSTORMS NXT –G Programming Guide.* Apress:USA