Setup Functions:

**robotType**
Sets the type of Robot to be used. This command will configure all of the default motor and sensor ports for the specified robot type.

Command:

```c
setRobotType(robotType);
```

Parameters: robotType

Valid Robot Types for robotType:
The type name of the robot to be used. Default value is 'VexIQClawbot'.

Usage without Parameters:

```c
robotType();
```
This snippet of code will set the robot type to VexIQClawbot by default.

Usage with Parameters:

```c
robotType(VexIQClawbot);
```
This snippet of code will set the robot type to VexIQClawbot. This will automatically set up the motor and sensor ports to match those of a default Vex IQ Clawbot. Clawbot requires the drive motors to be connected to Ports 1 & 6.

---

**setMotorReversed**
Sets the "reversed" flag of a motor to indicate that it’s direction should be reversed in relation to motor speed vs. rotation direction.

Command:

```c
setMotorReversed(motorPort, reversedValue);
```

Parameters: motorPort, reversedValue

Valid Robot Types for motorPort:
Name of motor to be controlled by the command. No default value – user must provide a motor name.

Valid Robot Types for reversedValue:
A Boolean value of "true" or "false" to indicate if the motor should be reversed or not.

Usage:

```c
setMotorReversed(left, true);
```
This snippet of code will set motor named "left" to be reversed.
Simple Movement Functions:

Note that for desirable results with the following set of functions, you must use the "robotType()"; The Function sets ‘VexIQClawbot’ by default. You should set your desired type in the beginning of your “task main()”.

**backward** - *The robot's drive motors must be connected to Port 1 and Port 6 for the Backward Natural Language command to work.* Moves your robot in the reverse (backwards) direction for a specific distance/time at a specific speed. The robot will come to a stop after it completes its movement.

**Command:**

```
backward(quantity, unitType, speed);
```

**Parameters:** quantity, unitType, speed

**Acceptable quantity values:**
The number of "unitType" for the robot to move - Any decimal (1.23) number is valid. There is no default value for 'quantity'.

**Acceptable unitType values:**
The type of units used for the quantity. Acceptable values: 'rotations', 'degrees', 'seconds', 'milliseconds', 'minutes'. Default value of 'rotations'.

**Valid range for speed:**
The speed (in percent) the robot will travel (range: 0 - 100). Default value of '50'

**Usage:**

```
backward(3.25, rotations, -75);
```

---

**forward** - *The robot's drive motors must be connected to Port 1 and Port 6 for the Forward Natural Language command to work.* Moves your robot in the forward direction for a specific distance/time at a specific speed. The robot will come to a stop after it completes its movement.

**Command:**

```
forward(quantity, unitType, speed);
```

**Parameters:** quantity, unitType, speed

**Acceptable quantity values:**
The number of "unitType" for the robot to move - Any decimal (1.23) number is valid. There is no default value for 'quantity'.

**Acceptable unitType values:**
The type of units used for the quantity. Acceptable values: 'rotations', 'degrees', 'seconds', 'milliseconds', 'minutes'. Default value of 'rotations'.

**Valid range for speed:**
The speed (in percent) the robot will travel (range: 0 - 100). Default value of '50'

**Usage:**

```
forward(3.25, rotations, 75);
```

Note that for desirable results with the following set of functions, you must use the "robotType()"; The Function sets ‘VexIQClawbot’ by default. You should set your desired type in the beginning of your “task main()”.

This snippet of code will move the robot backward for 3.25 rotations at 75 speed. The motor ports are determined by the use of setRobotType(robotType) function.

This snippet of code will move the robot forward for 3.25 rotations at 75 speed. The motor ports are determined by the use of setRobotType(robotType) function.
**turnLeft** - The robot's drive motors must be connected to Port 1 and Port 6 for the turnLeft Natural Language command to work. Rotates your robot to the left using one of two different types of turns (point turns and swing turns). The robot will come to a stop after it completes its movement.

![Robot facing left with turnLeft command]

**Command:**

```c
turnLeft(quantity, unitType, turnType, speed);
```

**Parameters:** quantity, unitType, turnType, speed

**Acceptable quantity values:**
The number of "unitType" for the robot to move - Any decimal (1.23) number is valid. There is no default value for 'quantity'.

**Acceptable unitType values:**
The type of units used for the quantity. Acceptable values: 'rotations', 'degrees', 'seconds', 'milliseconds', 'minutes'. Default value of 'rotations'.

**Acceptable values for turnType:**
The type of left turn the robot will make – ‘point’ turns will cause the right motor(s) to move forward and the left motor(s) to reverse. ‘swing’ turns will cause the right motor(s) to move forward and the left motor(s) will power off.

**Valid range for speed:**
The speed (in percent) the robot will travel (range: 0 - 100). Default value of '50'

**Usage:**

```c
forward(3.25, rotations, 60);
turnLeft(0.50, rotations, point, 50);
```

---

**turnRight** - The robot's drive motors must be connected to Port 1 and Port 6 for the turnRight Natural Language command to work. Rotates your robot to the right using one of two different types of turns (point turns and swing turns). The robot will come to a stop after it completes its movement.

![Robot facing right with turnRight command]

**Command:**

```c
turnRight(quantity, unitType, turnType, speed);
```

**Parameters:** quantity, unitType, turnType, speed

**Acceptable quantity values:**
The number of "unitType" for the robot to move - Any decimal (1.23) number is valid. There is no default value for 'quantity'.

**Acceptable unitType values:**
The type of units used for the quantity. Acceptable values: 'rotations', 'degrees', 'seconds', 'milliseconds', 'minutes'. Default value of 'rotations'.

**Acceptable values for turnType:**
The type of right turn the robot will make – ‘point’ turns will cause the left motor(s) to move forward and the right motor(s) to reverse. ‘swing’ turns will cause the left motor(s) to move forward and the right motor(s) will power off.

**Valid range for speed:**
The speed (in percent) the robot will travel (range: 0 - 100). Default value of '50'

**Usage:**

```c
forward(3.25, rotations, 75);
turnRight(0.50, rotations, point, 50);
```
Looping and Program Flow:

**repeat (forever)**
Repeats a block of code forever (infinite loop)

```plaintext
repeat(forever);
```

**Parameters:** N/A

**Usage:**

```
repeat(forever) {
  forward(3.25, rotations, 75);
  turnLeft(0.50, rotations, point, 50);
}
```

This snippet of code will move the robot forward for 3.25 rotations at 75 speed, and then perform a left point turn for .5 rotations at 50 speed. It will perform these two behaviors forever.

---

**repeat (numberOfLoops)**
Repeats a block of code for a specified number of loops.

```plaintext
repeat(numberOfLoops);
```

**Parameters:** numberOfLoops

**Acceptable numberOfLoops values:**
The number of "numberOfLoops" for the robot to move - Any integer (i.e. 1000) number is valid. There is no default value for 'number of loops'.

**Usage:**

```
repeat(4) {
  forward(2.5, rotations, 55);
  turnRight(1, rotations, point, 30);
}
```

This snippet of code will move the robot forward for 2.5 rotations at 55 speed, and then perform a left point turn for 1 rotation at 30 speed. It will repeat these behaviors 4 times.
repeatUntil (condition)
Repeats a block of code until a condition is true (loops until the condition is met).

```
X condition

Command:
repeatUntil(condition);
```

Parameters: A condition (see example for usage)
Usage:

```
encoderL = 0;
repeatUntil(SensorValue[encoderL]>200)
{
    forward(4, rotations, 60);
    turnLeft(0.35, rotations, point, 40);
}
```

This snippet of code will reset the encoder named "encoderL", move the robot forward for 4 rotations at 60 speed, and then perform a left point turn for .35 rotations at 40 speed. It will perform these two behaviors until the left encoder reads greater than 200.
Remote Control:

**arcadeControl** - The robot's drive motors must be connected to Port 1 and Port 6 for the **arcadeControl** Natural Language command to work. Remote control of your robot by using a single joystick (X and Y). This command must be inside of a loop in order to work.

**Command:**

```c
arcadeControl(verticalJoystick, horizontalJoystick, threshold);
```

**Parameters:** verticalJoystick, horizontalJoystick, threshold

**Acceptable verticalJoystick values:**
Joystick axis that determines speed. Default value of ‘ChA’.

**Acceptable horizontalJoystick values:**
Joystick axis that determines mix (rotation direction). Defaults value of ‘ChB’.

**Valid threshold range:**
Minimum joystick value required to move the robot – used for joysticks that do not return to perfect center to prevent drift. Default value of ‘10’.

Usage without parameters:

```
repeatUntil(forever)
{
    arcadeControl();
}
```

This snippet of code will enable arcade control forever. The vertical joystick will use the ‘ChA’ axis while the horizontal joystick will use the ‘ChB’ axis. The minimum threshold required to move the robot will be 10.

Usage with parameters:

```
repeatUntil(forever)
{
    arcadeControl(ChA, ChC, 20);
}
```

This snippet of code will enable arcade control forever. The vertical joystick will use the ‘ChA’ axis while the horizontal joystick will use the ‘ChC’ axis. The minimum threshold required to move the robot will be 20.
armControl
Remote control of a motor on your robot by using wireless controller buttons. This command must be inside of a loop in order to work.

Command:

```c
armControl(armMotorPort, upButton, downButton, armSpeed);
```

Parameters: armMotorPort, upButton, downButton, armSpeed

**Acceptable armMotorPort values:**
Name of motor to be controlled by the command. No default value – user must provide a motor name.

**Acceptable upButton values:**
The wireless controller button to make the motor spin clockwise (forward) when pressed. Default value of 'BtnRUp'.

**Acceptable downButton values:**
The wireless controller button to make the motor spin counterclockwise (reverse) when pressed. Default value of 'BtnRDown'.

**Valid armSpeed range:**
The speed the motor will travel. When the 'upButton' is pressed, the motor will travel the specified speed. When the 'downButton' is pressed, the motor will travel the negative value of the specified speed. Default value of '75'.

Usage with parameters:

```c
repeatUntil(forever)
{
  armControl(arm, BtnLUp, BtnLDown, 30);
}
```

This snippet of code will enable arcade control forever. The 'upButton' will use the 'L' Up button while the 'downButton' will use the 'L' Down button. The arm will move at 30 speed.
tankControl - The robot's drive motors must be connected to Port 1 and Port 6 for the tankControl Natural Language command to work. Remote control of your robot by using two joystick axes (Y and Y). This command must be inside of a loop in order to work.

Command:

```c
tankControl(rightJoystick, leftJoystick, threshold);
```

**Parameters:** rightJoystick, leftJoystick, threshold

- **Acceptable rightJoystick values:** Joystick axis that determines the speed of the right side of the robot. Defaults value of 'ChD'.
- **Acceptable leftJoystick values:** Joystick axis that determines the speed of the left side of the robot. Defaults value of 'ChA'.
- **Valid threshold range:** Minimum joystick value required to move the robot – used for joysticks that do not return to perfect center to prevent drift. Default value of '10'.

**Usage without parameters:**

```c
repeatUntil(forever)
{
    tankControl();
}
```

This snippet of code will enable tank control forever. The right joystick will use the 'ChD' axis while the left joystick will use the 'ChA' axis. The minimum threshold required to move the robot will begin at 10.

**Usage with parameters:**

```c
repeatUntil(forever)
{
    tankControl(ChC, ChB, 20);
}
```

This snippet of code will enable tank control forever. The right joystick will use the 'ChC' axis while the left joystick will use the 'ChB' axis. The minimum threshold required to move the robot will begin at 20.