```
#pragma config(Sensor, in1,    potent,        sensorPotentiometer)
#pragma config(Sensor, dgtl1,  limSwitch,     sensorDigitalIn)
#pragma config(Motor,  port2,            rightFront,   tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port3,            liftMotor,    tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port4,            scoopRight,   tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port5,            rightBack,    tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port6,            leftFront,    tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,            scoopLeft,    tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port8,            leftBack,     tmotorVex393_MC29, openLoop, reversed)
#pragma config(Motor,  port9,            liftMotor2,   tmotorVex393_MC29, openLoop)
#pragma platform(VEX)

//Competition Control and Duration Settings
#pragma competitionControl(Competition)
#pragma autonomousDuration(15)
#pragma userControlDuration(105)

#include "Vex_Competition_Includes.c"   //Main competition background code...do not modify!

const short leftButton = 1;
const short centerButton = 2;
const short rightButton = 4;

void waitForPress()
{
  while(nLCDButtons == 0){}
  wait1Msec(5);
}

void waitForRelease()
{
  while(nLCDButtons != 0){}
  wait1Msec(5);
}

void pre_auton()
{
  bStopTasksBetweenModes = true;
}
```

```c
void Reset()
{
  motor[scoopLeft] = 0;
  motor[scoopRight] = 0;
  motor[liftMotor] = 0;
  motor[liftMotor2] = 0;
  motor[leftBack] = 0;
  motor[leftFront] = 0;
  motor[rightFront] = 0;
  motor[rightBack] = 0;
  clearTimer(T1);
}

void driveMotorsFullSpeedForward()
{
  motor[leftBack] = 127;
  motor[leftFront] = 127;
  motor[rightFront] = 127;
  motor[rightBack] = 127;
}

void driveMotorsFullSpeedReverse()
{
  motor[leftBack] = -127;
  motor[leftFront] = -127;
  motor[rightFront] = -127;
  motor[rightBack] = -127;
}

void turnLeft()
{
  motor[leftBack] = 127;
  motor[leftFront] = 127;
}

void turnRight()
{
  motor[rightBack] = 127;
  motor[rightFront] = 127;
}
```

```
void liftLower()
{
  motor[liftMotor] = 127;
  motor[liftMotor2] = -127;
}

void liftRaise()
{
  motor[liftMotor] = -127;
  motor[liftMotor2] = 127;
}

void scoopRaiseHalfSpeed()
{
  motor[scoopLeft] = -63;
  motor[scoopRight] = 63;
}

void scoopLowerHalfSpeed()
{
  motor[scoopLeft] = 63;
  motor[scoopRight] = -63;
}

void scoopRaiseFullSpeed()
{
  motor[scoopLeft] = -127;
  motor[scoopRight] = 127;
}

void scoopLowerFullSpeed()
{
  motor[scoopLeft] = 127;
  motor[scoopRight] = -127;
}

void RedWithoutBucky()
{
    bool lift = true;
```

```
//
//          RED WITHOUT BUCKY
//          AUTONOMOUS
//
clearTimer(T1);

//Lower scoop in beginning to obtain ball later
while (time1[T1] < 500)
{
  scoopLowerFullSpeed();
}
//Reset motors and clear Timer1
Reset();

//Move forward and get scoop positioned under the ball and next to buckyballs on the bump
while(time1[T1] < 1325)
{
  driveMotorsFullSpeedForward();
}
//Reset motors and clear Timer1
Reset();

//Raise the scoop completely and gain complete hold of the large ball
while (time1[T1] < 350)
{
  scoopRaiseHalfSpeed();
}
//Reset motors and clear Timer1
Reset();

//Turn using the right side going forward and knock off the three buckyballs on the bump
while (time1[T1] < 350)
{
  turnRight();
}
//Reset motors and clear Timer1
Reset();

//Drive backwards to have the lift hooks positioned perfectly over the bar.
while (time1[T1] < 1450)
```

```
  {
    driveMotorsFullSpeedReverse();
  }
  //Reset motors and clearTimer1
  Reset();

  //Lower the lift to high hang with a ball
  while(lift)
  {
    liftLower();
    if (SensorValue[dgtl1] == 0){
      lift = false;
    }
  }
  //Stop the motors once it is actually pressed
  Reset();
}

void BlueWithoutBucky()
{
  bool lift = true;
  clearTimer(T1);
  //
  //           BLUE WITHOUT BUCKY
  //           AUTONOMOUS
  //
  //Lower scoop in beginning to obtain ball later
  while (time1[T1] < 500)
  {
    scoopLowerFullSpeed();
  }
  //Reset motors and clear Timer1
  Reset();

  //Move forward and get scoop positioned under the ball and next to buckyballs on the bump
  while(time1[T1] < 1225)
  {
    driveMotorsFullSpeedForward();
  }
  //Reset motors and clear Timer1
```

```
  Reset();

  //Raise the scoop a little bit to gain better hold of the large ball
  while (time1[T1] < 350)
  {
    scoopRaiseFullSpeed();
  }
  //Reset motors and clear Timer1
  Reset();

  //Turn using the left side going forward and knock off the three buckyballs on the bump
  while (time1[T1] < 450)
  {
    turnLeft();
  }
  //Reset motors and clear Timer1
  Reset();

  //Drive backwards to have the lift hooks positioned perfectly over the bar.
  while (time1[T1] < 1350)
  {
    driveMotorsFullSpeedReverse();
  }
  //Reset motors and clearTimer1
  Reset();

  //Lower the lift to high hang with a ball
  while(lift)
  {
    liftLower();
    if (SensorValue[dgtl1] == 0){
      lift = false;
    }
  }
  //Reset motors anc clear Timer1
  Reset();
}

void BlueAutonomous()
{
```

```
bool lift = true;
clearTimer(T1);
//
//              BLUE
//              AUTONOMOUS
//
//Lower scoop in beginning to obtain ball later
while (time1[T1] < 500)
{
   scoopLowerFullSpeed();
}
//Reset motors and clear Timer1
Reset();

//Move forward and get scoop positioned under the ball and next to buckyballs on the bump
while(time1[T1] < 1225)
{
   driveMotorsFullSpeedForward();
}
//Reset motors and clear Timer1
Reset();

// Move back 100 milliseconds to get in line better with the buckyballs to knock them off
// and to help with allignment with the bar later on
while (time1[T1] < 100)
{
   driveMotorsFullSpeedReverse();
}
//reset motors and clear Timer1
Reset();

//Raise the scoop a little bit to gain better hold of the large ball
while (time1[T1] < 150)
{
   scoopRaiseFullSpeed();
}
//Reset motors and clear Timer1
Reset();

//Turn using the left side going forward and knock off the three buckyballs on the bump
```

```
  while (time1[T1] < 550)
  {
    turnLeft();
  }
  //Reset motors and clear Timer1
  Reset();
  //Raise the scoop completely and gain complete hold of the large ball
  while (time1[T1] < 150)
  {
    scoopRaiseFullSpeed();
  }
  //Reset motors and clear Timer1
  Reset();

  //Drive backwards to have the lift hooks positioned perfectly over the bar.
  while (time1[T1] < 1350)
  {
    driveMotorsFullSpeedReverse();
  }
  //Reset motors and clearTimer1
  Reset();

  //Lower the lift to high hang with a ball
  while(lift)
  {
    liftLower();
    if (SensorValue[dgtl1] == 0){
      lift = false;
    }
  }
  //Reset motors anc clear Timer1
  Reset();
}

void RedAutonomous()
{
  bool lift = true;
  //
  //              RED
  //              AUTONOMOUS
```

```c
//
clearTimer(T1);

//Lower scoop in beginning to obtain ball later
while (time1[T1] < 500)
{
  scoopLowerFullSpeed();
}
//Reset motors and clear Timer1
Reset();

//Move forward and get scoop positioned under the ball and next to buckyballs on the bump
while(time1[T1] < 1325)
{
  driveMotorsFullSpeedForward();
}
//Reset motors and clear Timer1
Reset();

//Raise the scoop a little bit to gain better hold of the large ball
while (time1[T1] < 100)
{
  scoopRaiseHalfSpeed();
}
//Reset motors anc clear Timer1
Reset();

// Move back 100 milliseconds to get in line better with the buckyballs to knock them off
// and to help with allignment with the bar later on
while (time1[T1] < 100)
{
  driveMotorsFullSpeedReverse();
}
//reset motors and clear Timer1
Reset();



//Turn using the right side going forward and knock off the three buckyballs on the bump
while (time1[T1] < 550)
```

```
  {
    turnRight();
  }
  //Reset motors and clear Timer1
  Reset();

  //Raise the scoop completely and gain complete hold of the large ball
  while (time1[T1] < 350)
  {
    scoopRaiseHalfSpeed();
  }
  //Reset motors and clear Timer1
  Reset();

  //Drive backwards to have the lift hooks positioned perfectly over the bar.
  while (time1[T1] < 1550)
  {
    driveMotorsFullSpeedReverse();
  }
  //Reset motors and clearTimer1
  Reset();

  //Lower the lift to high hang with a ball
  while(lift)
  {
    liftLower();
    if (SensorValue[dgtl1] == 0){
      lift = false;
    }
  }
  //Stop the motors once it is actually pressed
  Reset();
}

task autonomous()
{
  int count = 0;

  clearLCDLine(0);
  clearLCDLine(1);
```

```c
  while(nLCDButtons != centerButton){
    switch(count){
    case 0:
      //Display first choice
      displayLCDCenteredString(0, "Blue");
      displayLCDCenteredString(1, "<     Enter    >");
      waitForPress();
      //Increment or decrement "count" based on button press
      if(nLCDButtons == leftButton)
      {
        waitForRelease();
        count = count - 1;
      }
      else if(nLCDButtons == rightButton)
      {
        waitForRelease();
        count = count + 1;
      }
      break;
    case 1:
      //Display second choice
      displayLCDCenteredString(0, "Red");
      displayLCDCenteredString(1, "<     Enter    >");
      waitForPress();
      //Increment or decrement "count" based on button press
      if(nLCDButtons == leftButton)
      {
        waitForRelease();
        count = count - 1;
      }
      else if(nLCDButtons == rightButton)
      {
        waitForRelease();
        count = count + 1;
      }
      break;
    case 2:
      //Display second choice
      displayLCDCenteredString(0, "Red Without Bucky");
      displayLCDCenteredString(1, "<     Enter    >");
```

```c
    waitForPress();
    //Increment or decrement "count" based on button press
    if(nLCDButtons == leftButton)
    {
      waitForRelease();
      count = count - 1;
    }
    else if(nLCDButtons == rightButton)
    {
      waitForRelease();
      count = count + 1;
    }
    break;
    case 3:
    displayLCDCenteredString(0, "Blue Without Bucky");
    displayLCDCenteredString(1, "<       Enter      >");
    waitForPress();
      if(nLCDButtons == leftButton)
    {
      waitForRelease();
      count = count - 1;
    }
    else if(nLCDButtons == rightButton)
    {
      waitForRelease();
      count = count + 1;
    }
    break;
  default:
    count = 0;
    break;
  }
}
  switch(count){
  case 0:
    BlueAutonomous();
    break;
  case 1:
    RedAutonomous();
    break;
```

```c
    case 2:
      RedWithoutBucky();
      break;
    case 3:
      BlueWithoutBucky();
      break;
  }
}

task usercontrol()
{
  while (true)
  {


    //Joystick Movement
    motor[leftBack] = vexRT[Ch3];
    motor[leftFront] = vexRT[Ch3];
    motor[rightBack] = vexRT[Ch2];
    motor[rightFront] = vexRT[Ch2];


    //Joystick Digital Buttons using embedded if statements

    //Raise Scoop
    if (vexRT[Btn5U] == 1)
    {
      motor[scoopRight] = 80;
      motor[scoopLeft] = -80;
    }
    else
    {
      // Lower Scoop
      if (vexRT[Btn5D] == 1)
      {
        motor[scoopRight] = -80;
        motor[scoopLeft] = 80;
      }
      else
      {
```

```c
    if (vexRT[Btn8D] == 1)
    {
      motor[scoopLeft] = 45;
      motor[scoopRight] = -45;
    }
    else
    {
      if (vexRT[Btn8U] == 1)
      {
        motor[scoopRight] = 55;
        motor[scoopLeft] = -55;
      }
      else
      {
        //turn motors off if no buttons on channel 5 are pressed
        motor[scoopRight] = 0;
        motor[scoopLeft] = 0;
      }
    }
  }
}

//Lift Control
//Raise Lift
if (vexRT[Btn6U] == 1)
{
  motor[liftMotor] = -127;
  motor[liftMotor2] = 127;
}
else
{
  //Lower Lift
  if (vexRT[Btn6D] == 1)
  {
    motor[liftMotor] = 127;
    motor[liftMotor2] = -127;
  }
  else
  {
    // Turn motors off if no buttons are pressed on Channel 6
```

```
        motor[liftMotor] = 0;
        motor[liftMotor2] = 0;
      }
    }
  }
}
```