```
#pragma config(Sensor, in1,    YB,             sensorPotentiometer)
#pragma config(Sensor, in2,    lift_Pos,       sensorPotentiometer)
#pragma config(Sensor, in3,    turn_,          sensorNone)
#pragma config(Sensor, in4,    stacker_Pos,    sensorPotentiometer)
#pragma config(Sensor, in5,    claw_Pos,       sensorPotentiometer)
#pragma config(Sensor, in8,    turn_Deg,       sensorGyro)
#pragma config(Sensor, dgtl1,  drive_Dist,     sensorQuadEncoder)
#pragma config(Sensor, dgtl3,  mogo_In,        sensorTouch)
#pragma config(Sensor, dgtl4,  cone_Detect,    sensorSONAR_cm)
#pragma config(Motor,  port1,           MOGORIGHT,
tmotorVex393_HBridge, openLoop)
#pragma config(Motor,  port2,           RIGHTLIFT,
tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port3,           BARLIFT,       tmotorVex393_MC29,
openLoop)
#pragma config(Motor,  port4,           BACKLEFT,
tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port5,           FRONTLEFT,
tmotorVex393HighSpeed_MC29, openLoop)
#pragma config(Motor,  port6,           FRONTRIGHT,
tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port7,           BACKRIGHT,
tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port9,           LEFTLIFT,
tmotorVex393HighSpeed_MC29, openLoop, reversed)
#pragma config(Motor,  port10,          MOGOLEFT,
tmotorVex393_HBridge, openLoop, reversed)
//*!!Code automatically generated by 'ROBOTC' configuration wizard
!!*//

void moveBase(int speed)
{
      motor[FRONTRIGHT]      =     speed;
      motor[FRONTLEFT] =     speed;
      motor[BACKRIGHT] =     speed;
      motor[BACKLEFT]  =     speed;
}

void forwardsPID(int dist)
{
      float Kp = 0.09;
      float Kd = 0.0;
      float Proportion;
      float Derivative;
      float finalSpeed;
      int error;

      //clear encoder
      SensorValue[BACKLEFT] = 0;

      while (true)
{
      //calculate the error
      error = dist - SensorValue[BACKLEFT];
```

```
	//calculate the proportion
	Proportion = error *Kp;

	//Cap on proportion so, it doesn't send ridiculous values
	if (Proportion > 127) Proportion = 127.0;

	//calculate final speed to send to the motors
	finalSpeed = Proportion;

	//send the final speed to the motors
	moveBase(finalSpeed);
	wait1Msec(20);
}

}


task main()
{
	forwardsPID(1000);
}
```