Moon Rover Assignment#1

The following code will allow the robot follow a predetermined path and knock an “alien” off its block.

```
task main()
{
 motor[motorB] = 95;              //Forward movement in both motors for 4 seconds.
 motor[motorC] = 95;
 wait1Msec(4000);

 motor[motorB] = 0;                          //A left turn for 0.8 of a second.
 motor[motorC] = 100;
 wait1Msec(800);

 motor[motorB] = 100;                        //Forward movement for 1.1 seconds.
 motor[motorC] = 100;
 wait1Msec(1100);

 motor[motorB] = 100;                        //A right turn for 0.8 of a second.
 motor[motorC] = 0;
 wait1Msec(800);

 motor[motorB] = 95;                         //Forward movement for 1.9 seconds.
 motor[motorC] = 95;
 wait1Msec(1900);

 motor[motorB] = 100;                        //A right turn for 0.8 of a second.
 motor[motorC] = 0;
 wait1Msec(800);

 motor[motorB] = 95;                         //Forward movement for 2.75 seconds.
 motor[motorC] = 95;.
 wait1Msec(2750);

 motor[motorB] = 0;                          //A left turn for 0.55 of a second.
 motor[motorC] = 95;
 wait1Msec(550);

 motor[motorB] = 90;                  //Forward movement on a diagonal for 1.5 seconds.
  motor[motorC] = 100;
  wait1Msec(1500);

  motor[motorB] = 0;                         //Turning in a counterclockwise motion for 3
  motor[motorC] = 100;                         seconds in order to  knock the alien off./.
  wait1Msec(3000);

 }
```

Bumper Bot Assignment#2

This code will direct the robot to move forward until the touch sensor is pressed. Then it will reverse, make a 90 degree right turn and will continue with forward motion. This program runs on an infinite loop.

```
task main()
{
        while(true)                    //Loop Program…forever!
        {
                motor[motorB] = 95;                 //Forward movement
                motor[motorC] = 95;
                if(SensorValue[S1] == 1)

                {
                        motor[motorB] = -95;  //Backwards movement
                        motor[motorC] = -95;
                        wait1Msec(2000);
                        motor[motorB] = 100;  //Right turn for 0.9 of a second.
                        motor[motorC] = 0;
                        wait1Msec(900);

                }
        }
}
```

LOOP!

```
while(true)
{
}       Whatever is in the brackets
        Will repeat forever
```

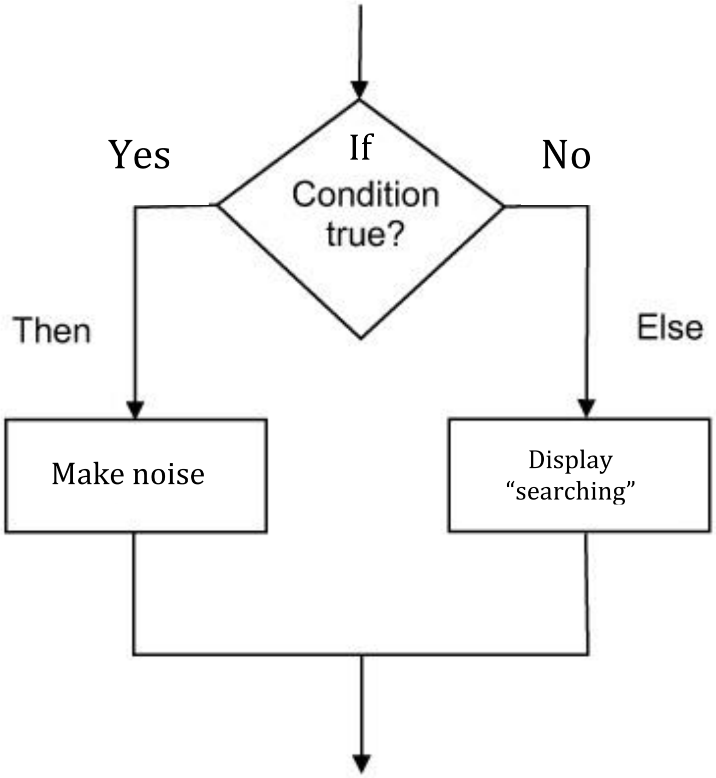
Searching for Gold Assignment#3

This code directs the robot to beep whenever it senses the color black. It runs on a continuous loop and includes an "if" statement that makes the NXT beep continuously **if** the light sensor is below 20.

```
task main()

{
                wait1Msec(3000);                                //Wait 3 seconds to start.

                while(true)                                      // Run continuously.

                {
                        nxtDisplayCenteredTextLine(2, " %d ", SensorValue(S1));
                        nxtDisplayCenteredTextLine(4, "Searching");
                        nxtDisplayCenteredTextLine(5, "For Black.");

                        if(SensorValue[S1]<20)   // When sensor value is less than 20

                        {
                                playTone(780, 15);              //Play noise.
                                wait1Msec(500);        //Wait 0.5 of a second.


                        }
                }
}
```

"if statements"

An "*if*" statements allows a computer to **make decisions.** *if* statements **don't repeat like a while loop**, they just get the computer to do *one thing* or *another* based on some sort of information. Look at the syntax and example below:



Line Following Assignment#4

The following program will have the robot follow a black line. When the color sensor sees black it turns *away* from the line, when it sees white it turns *towards* the line. The robot also has a sonar sensor attached to it that will make a sound and stop robot when an object blocks its path.

```
task main()
{
        while(true)   //loops forever
        {

                while(SensorValue[S1]>20)   //loops while S1>20
                {
                        displayBigTextLine(2,"%d",SensorValue[S4]);

                        if (SensorValue[S4] > 30)  //if light greater than 30
                        {
                                motor[motorC] = 80;
                                motor[motorB] = 0;   // turn left
                                wait1Msec(35);
                        }
                        else
                        {
                                motor[motorC] = 0;     //otherwise turn right
                                motor[motorB] = 80;
                                wait1Msec(35);
                        }
                }
                motor[motorC] = 0;      //sensor 1 <20, stop motors and play sound
                motor[motorB] =0;
                playSoundFile("Woops.rso");
                wait1Msec(200);
        }
}
```

Area Finder Code#5

```
int dist1;      //Variable Distance One
int dist2;      //Variable Distance Two
int total;      //Variable total of both distances multiplied by each other
task main()
{
        eraseDisplay();
        while(SensorValue(S1)==0)       //While button not pressed
        {
         nxtDisplayCenteredTextLine(2,"Finding");
        nxtDisplayCenteredTextLine(3,"Dist1");

        }
        eraseDisplay();                                 //Erase display
        while(SensorValue[S1]==1)          //While sensor value is one
        {

                dist1=SensorValue(S4);  //put sonar value in dist1
                wait1Msec(500);         //Wait 0.5 seconds
                playSound(soundBeepBeep);               //Plays sound “Beep Beep”
                nxtDisplayCenteredBigTextLine(2,"%d",SensorValue[S4]);          //Display
        }
        eraseDisplay();                                 //Erase display
        while(SensorValue[S1]==0)               //While sensor value is zero the following will
        {

                nxtDisplayCenteredTextLine(2,"Finding");        //Displays on screen “Finding”
                nxtDisplayCenteredTextLine(4,"Dist2");          //Displays on screen “Dist2”

        }
        eraseDisplay();                                         //Erase display
        while(SensorValue(S1)==1)               //While sensor value is one the following will
        {

                dist2=SensorValue(S4);          //Sonar sensor finds distance 2
                wait1Msec(500);                 //Wait 0.5 of a second
                nxtDisplayCenteredBigTextLine(2,"%d",SensorValue[S4]);          //Display
                playSound(soundBeepBeep);                               //Plays sound “Beep
        }
        eraseDisplay();                                         //Erase Display
        total=dist1*dist2;              //Finds total by multiplying distances
        wait1Msec(500);                 //Waits 0.5 seconds
        nxtDisplayCenteredBigTextLine(3,"%d",total);    //Displays total on screen
        wait1Msec(1500);                                //Waits 1.5 seconds
        eraseDisplay();                                 //Erase display
        nxtDisplayCenteredTextLine(3,"Finished");       //Display “Finished” on screen
        wait1Msec(3000);                                //Waits 3 seconds
}
```

Line Counter#6

```
int count;
task main ()
{
     wait1Msec(1000);
     count = 0;
     eraseDisplay();

     while (nMotorEncoder[motorB]<1050)
     {
          motor[motorB] = 30;
          motor[motorC] = 30;

          if(SensorValue[S1]<16)
          {
          count=count+1;
          nxtDisplayCenteredBigTextLine (3, "%d", count);
          wait1Msec(100);
          nMotorEncoder[motorB] = 0;
          playSound(soundDownwardTones);

          }
     }

     eraseDisplay();
     nxtDisplayCenteredTextLine(3, "the count was");
     nxtDisplayCenteredTextLine(4, "%d", count);
     wait1Msec(3000);
}
```

Radar Gun#7

```
float d1;
float d2;
float speed;
float clock;
task main()
{
      while(SensorValue[S1]== 0)
      {
            displayCenteredTextLine(3, "Press the button");
            displayCenteredTextLine(4, "to begin");

      }
      eraseDisplay();
      while(SensorValue[S1]==1)
      {
            d1=(SensorValue[S2]);      //collects first distance
            clearTimer(T1);            //clears timer

      }
      while(SensorValue[S1]== 0)
      {
            displayCenteredTextLine(3, "dist1 complete");

      }
      eraseDisplay();
      while(SensorValue[S1]==1)
      {
            d2=(SensorValue[S2]);     //collects second distance
            clock=time1(T1);  //puts T1 time into variable called: clock

      }

      speed=(d1-d2)/clock;

      nxtDisplayCenteredTextLine(3,"%1.2f", speed);
      wait1Msec(1500);
      eraseDisplay();
      nxtDisplayCenteredTextLine(3,"speed was",);
      nxtDisplayCenteredTextLine(4,"%1.2f", speed);
      wait1Msec(2000);


}
```

Gate Counter#8

```
int count;
task main()
{
 count=0;
 nMotorEncoder[motorC] = 0;  // reset motorEncoder to zero
 while(count<10)
 {
  if(nMotorEncoder[motorC]>45)
  {
   wait1Msec(1000);
   motor[motorC] = -15;
   wait1Msec(630);
   motor[motorC] = 0;
   wait1Msec(400);
   count=count+1;
   nxtDisplayCenteredBigTextLine(3,"%d",count);
   wait1Msec(500);
   nxtDisplayCenteredBigTextLine(3,"%d",count);
  }
 }
            // This second section is for bonus!!!

 while(true)
 {
  motor[motorC] = 0;

     if(nMotorEncoder[motorC]>3)
     {
     motor[motorC] = -60;
     nxtDisplayCenteredTextLine(3,"Scan Pass");
     }

     if(SensorValue[S2]<17)
     {
     motor[motorC] = 15;
     wait1Msec(670);
     motor[motorC] = 0;
     wait1Msec(650);
     motor[motorC] = -15;
      wait1Msec(70);
     count=count+1;
     nxtDisplayCenteredBigTextLine(3,"%d", count);
     wait1Msec(600);
     }
  }
}
```

Arrays

```
char letter;
int b;
char name[9];
int i;

task main()
{

     i=0;
     letter=65;
     nxtDisplayTextLine(3,"scroll to view");
     nxtDisplayTextLine(4,"orange to select");
     nxtDisplayTextLine(5,"> to view word");
     wait1Msec(5000);
     eraseDisplay();
     while(nNxtButtonPressed!=1)
     {
          if(nMotorEncoder[motorC]>30)
          {
               letter=letter+1;
               nxtDisplayCenteredBigTextLine(3,"%c",letter);
               nMotorEncoder[motorC]=0;
          }
          if(nMotorEncoder[motorC] < - 30)
          {
               letter=letter-1;
               nxtDisplayCenteredBigTextLine(3,"%c",letter);
               nMotorEncoder[motorC]=0;

          }
          if(nNxtButtonPressed==3)
          {
               wait1Msec(100);
               name[i]=letter;
               i=i+1;
               wait1Msec(2000);
          }
     }

          i=0;
```

```
    while(true)
    {
        playSound(soundDownwardTones);

        eraseDisplay();
        while(i<9)
        {
            nMotorEncoder[motorC]=0;
            nxtDisplayBigStringAt(2+(13*i),40,"%c",name[i]);
            i=i+1;
            wait1Msec(1000);
        }

        eraseDisplay();

        if(nNxtButtonPressed==2)
        {

            playSound(soundDownwardTones);
            b=0;
            eraseDisplay();
            while(i>0)
            {
            nMotorEncoder[motorC]=0;
            nxtDisplayBigStringAt(61-(13*b),40,"%c",name[i]);
            b=b+1;
            wait1Msec(1000);
            }
        }
    }
}
```

Drawing Example Program

```
task main()
{
 while(true)      //loops program
{
nxtDrawCircle(77, 57 ,14); //draws circle in "sky"
nxtDrawRect(5, 26, 17, 3); //draws rectangle on leftside
nxtDrawRect(17, 18, 29, 3); //draws rectangle next to previous
nxtDrawRect(29, 23, 41, 3); //draws rectangle next to previous
nxtDrawRect(41, 35, 53, 3); //draws rectangle slightly away from previous
nxtDrawRect(63, 35, 75, 3); //draws rectangle next to previous
nxtDrawRect(75, 41, 87, 3); //draws rectangle next to previous
nxtDrawRect(87, 23, 99, 3); //draws rectangle next to previous
nxtDrawLine(1, 3, 99, 3); //draws line at bottom of screen
nxtSetPixel(23, 46);               //draws a bird
nxtSetPixel(24, 47);
nxtSetPixel(25, 48);
nxtSetPixel(26, 47);
nxtSetPixel(27, 47);
nxtSetPixel(28, 48);
nxtSetPixel(29, 47);
nxtSetPixel(30, 46); //finishes bird
nxtSetPixel(32, 43);//draws another bird
nxtSetPixel(33, 44);
nxtSetPixel(34, 45);
nxtSetPixel(35, 44);
nxtSetPixel(36, 44);
nxtSetPixel(37, 45);
nxtSetPixel(38, 46);
nxtSetPixel(39, 45);               //finishes second bird


}
}
```

Sample Animation Program

```
task sun()
{
        int mover;
        mover=0;

        while(mover<13)
        {
                nxtDrawRect(5, 26, 17, 3);
                nxtDrawRect(17, 18, 29, 3);
                nxtDrawRect(29, 23, 41, 3);
                nxtDrawRect(41, 35, 53, 3);
                nxtDrawRect(63, 35, 75, 3);
                nxtDrawRect(75, 41, 87, 3);
                nxtDrawRect(87, 23, 99, 3);

                nxtDrawEllipse(77, 37+mover, 91, 51+mover);
                wait1Msec(300);
                nxtEraseEllipse(77, 37+mover, 91, 51+mover);
                mover=mover+1;

                nxtDrawLine(1, 3, 99, 3);
        }
        nxtDrawRect(5, 26, 17, 3);
        nxtDrawRect(17, 18, 29, 3);
        nxtDrawRect(29, 23, 41, 3);
        nxtDrawRect(41, 35, 53, 3);
        nxtDrawRect(63, 35, 75, 3);
        nxtDrawRect(75, 41, 87, 3);
        nxtDrawRect(87, 23, 99, 3);

        nxtDrawEllipse(77, 37+mover, 91, 51+mover);
        wait1Msec(300);

}
task bird1()
{
        int mover2;
        mover2=0;
        while (true)

        {
                nxtSetPixel(23, 46+mover2);
                nxtSetPixel(24, 47+mover2);
                nxtSetPixel(25, 48+mover2);
                nxtSetPixel(26, 47+mover2);
                nxtSetPixel(27, 47+mover2);
                nxtSetPixel(28, 48+mover2);
                nxtSetPixel(29, 47+mover2);
                nxtSetPixel(30, 46+mover2);
                wait1Msec(250);
                nxtClearPixel(23, 46+mover2-1);
                nxtClearPixel(24, 47+mover2-1);
                nxtClearPixel(25, 48+mover2-1);
                nxtClearPixel(26, 47+mover2-1);
                nxtClearPixel(27, 47+mover2-1);
                nxtClearPixel(28, 48+mover2-1);
                nxtClearPixel(29, 47+mover2-1);
                nxtClearPixel(30, 46+mover2-1);
                mover2=mover2+1;
        }
}

task bird2()
{
        int mover3;
        mover3=0;
        while (true)
        {
                nxtSetPixel(32, 43+mover3);
                nxtSetPixel(33, 44+mover3);
                nxtSetPixel(34, 45+mover3);
                nxtSetPixel(35, 44+mover3);
                nxtSetPixel(36, 44+mover3);
                nxtSetPixel(37, 45+mover3);
                nxtSetPixel(38, 46+mover3);
                nxtSetPixel(39, 45+mover3);
                wait1Msec(250);
                nxtClearPixel(32, 43+mover3-1);
                nxtClearPixel(33, 44+mover3-1);
                nxtClearPixel(34, 45+mover3-1);
                nxtClearPixel(35, 44+mover3-1);
                nxtClearPixel(36, 44+mover3-1);
                nxtClearPixel(37, 45+mover3-1);
                nxtClearPixel(38, 46+mover3-1);
                nxtClearPixel(39, 45+mover3-1);
                mover3=mover3+1;
        }
}

task main()
{
        startTask(sun);
        startTask(bird1);
        startTask(bird2);
        while(true)
        {

        }
}
```