```
1. //Examples of basic coding essentials
2.
3. import //Used to import functions needed such as DCmotors and Gyroscopes
4.
5. class main(){
6.     public static void basics(){
7.
8.         //Variables first part ex. bool, int, etc. is the type of variable and the
   second part is the variable name that can be almost anything and is case sensative
9.         boolean tf=true; //Booleans hold a binary value either 1 (true) or 0 (false)
10.
11.        int whole=9; //Integers are whole numbers meaning that they can be positive or
   negative but do not have decimals or fractions, naturally they are real
12.
13.        float decimal=5.5; //Floats are the archaic form of doubles that can be any real
   number, floats are used on the controllers because that is the type of signal they send
   out
14.
15.        double bigDecimal=3.455432346342; //Doubles are more accurate floats and can be
   any real number
16.
17.        //NOTE can also be initialized as
18.        // double bigDecimal;
19.        // bigDecimal=3.455432346342;
20.
21.        if(whole==6){//This checks to see if the value of whole (which is 9) is equal to
   6, which because it is not, it is false and so does not execute what is in the braces
22.            //this is false
23.        }else if(whole>=10){//This checks to see if the value of whole (which is 9) is
   greater than or equal to 10 which, because it is not, it is false and so does not
   execute what is in the braces
24.            //this is false
25.        }else if(whole<=6){//This checks to see if the value of whole (which is 9) is
   less than or equal to 6 which, because it is not, it is false and so does not execute
   what is in the braces
26.            //this is false
27.        }else if(whole!=9){//This checks to see if the value of whole (which is 9) is
   not equal to 9 which, because it is equal to 9, it is false and so does not execute what
   is in the braces
28.            //this is false
29.        }else if(whole>9){//This checks to see if the value of whole (which is 9) is
   greater than 9 which, because it is not, it is false and so does not execute what is in
   the braces
30.            //this is false
31.        }else if(whole<10){//This checks to see if the value of whole (which is 9) is
   less than 10 which, because it is, it is true and so does execute what is in the braces
32.            //this is true
33.        }else{//This simply states that as long as none of the other statements are true
   then this will execute what is in the braces
34.            //Will occur as long as none of the others occur
35.        }
36.
37.        if(tf){//Because an if statement simply checks to see if what is in the
   parenthesis is true if you input true (which tf is true) into the parenthesis then what
   is in the braces will be executed
38.            //This is true
39.        }
```

```
40.
41.         int a=5;
42.         int b=2;
43.         //A more advanced comparison that can be used in anything if/while/for
    statements
44.         if(a=5 && b=5){//&& means (and) so it is checking if a=5 and b=5 which, because
    b is not equal to 5, the statement is false
45.             //This is false
46.         } else if(a=5 || b=5)//|| means (or) so it is checking if a=5 or b=5 which,
    because a is true even though b is false, the statement is true
47.
48.         int loop=0;
49.         while(loop < 100){//while loop simply states that while the value is false the
    loop will occur meaning that if you were to put false in the place of (loop<100) it
    would occur infinitely
50.             //placement of the action matters greatly in a loop, placed here and the
    action will occur 99 times but if the funtion was changed to <= it would occur 100 times
51.             loop+=1; //+=1 means the same as saying loop=loop+1
52.             //placement of the action matters greatly in a loop, placed here and the
    action will occur 100 times but if the function was changed to <= it would occur 101
    times
53.         }
54.
55.         for(int loops=0; loops<100; loops+=1;){// for loops are basically the same as
    while loops just different formatting (initialize variable; set arguement; set
    increment;)
56.             //This will occur 99 times however because of the same reason as is in the
    while loop if it were changed (loops<=100;) it would occur 100 times
57.         }
58.     }
59.
60.     public static void math(){
61.         int addition=3+7; //This would return 10 because 3+7=10
62.         int subtraction=7-3; //This would return 4 because 7-3=4
63.         int multiplication=5*3; //This would return 15 because 5*3=15
64.         double division=4/3; //This would return 1.33333333 because 4/3=1.33333333
65.         double remainder=4/3; //This would return 1 because the remainder of 4/3 is 1
    (basically the whole number that is left after you remove what can be equally divided
    out)
66.         double remainder2=7/3; //This would also return 1 because 3 can only go into 7
    twice and 7-6=1
67.         double remainder3=8/3; //But this would return 2 because 3 can only go into 8
    twice and 8-6=2
68.         double remainder4=10/3; //This would return 1 because 3 can only go into 10 3
    times and 10-9=1
69.         double remainder5=9/3; //This would return 0 because 3 goes into 9 without any
    remainders
70.     }
71.
72.     public static void complexities(){
73.
74.         //Variables - Arrays first you initialize the variable type ex. bool, int, etc
    and then add [] to make it an array. Then add the variable that can be named anything
    but is case sensative.
75.         //You then have to allocate the size of the array by using = new
    variabletype[number of variables]
76.         boolean[] boolArray;//Creates an array of booleans
```

```
77.         boolArray = new boolean[3];//Allocates the array size to 3 booleans
78.         boolArray[0]=true;//It starts from zero ALWAYS
79.         boolArray[1]=false;
80.         boolArray[2]=true;//Because of starting from zero the biggest number is always
    number allocated-1
81.
82.         int[] intArray;//Creates an array of integers
83.         intArray = new int[3];//Allocates the array size to 3 integers
84.         intArray[0]=50;
85.         intArray[1]=2;
86.         intArray[2]=5231;
87.
88.         float[] floatArray;//Creates an array of floats
89.         floatArray = new float[3];//Allocates the array size to 3 floats
90.         floatArray[0]=4.21;
91.         floatArray[1]=50.32;
92.         floatArray[2]=.23;
93.
94.         double[] doubleArray;//Creates an array of doubles
95.         doubleArray = new double[3];//Allocates the array size to 3 doubles
96.         doubleArray[0]=432.513;
97.         doubleArray[1]=3126.5235;
98.         doubleArray[2]=13276.36867;
99.
100.        //NOTE can also be configured as double[] doubleArray = new double[3]; just like
    how the original variables were initialized
101.
102.        if(boolArray[0]){
103.            //This is true
104.        }else if (boolArray[1]){
105.            //This is false
106.        }else if (boolArray[2]){
107.            //This is true
108.        }
109.
110.        for(int loops=0; loops<=2; loop+=1;){
111.            intArray[loops];//Because loops is a number it can be used in place of the
    variable number, this function would cause the values of intArray[] to be displayed in
    order (50,2,5231)
112.        }
113.
114.        int errors=6;
115.        double[] totalError = new float[errors];//This will create a double array with a
    size of 6 (0,1,2,3,4,5)
116.
117.        for(int loops=0; loops<=errors; loop+=1;){
118.            int currentError; //This could potentially be a function that uses the
    gyroscope plus the speed traveled to collect the amount of "error" or distance off path
119.            totalError[loops]=currentError;//Sets each "second" of error to the
    totalError
120.            netError+=totalError[loops];//Conbines every totalError[] into a cumulative
    netError
121.        }
122.
123.
124.    }
125. }
```