# IF-THEN functions in Inventor using SIGN function

In Inventor, how can you assign values to Y according to the following conditions without using user-defined VBA functions?

```
IF (X > 0 in) AND (X < 46 in) THEN
    Y = 15 in
ELSE IF (X >= 46 in) AND (X < 55 in) THEN
    Y = 17 in
ELSE IF (X >= 55 in) AND (X <= 60 in) THEN
    Y = 19 in
ELSE
    Y = 10 in
```

Inventor has a function called SIGN(..) which returns 1 (ul) if the argument is greater than zero, and 0 (ul) if the argument is less than or equal to zero. The argument may have units like in, mm or radian, or it may be unitless.

This SIGN function can be used to set up complex IF-THEN formulas. We may need to define intermediate Boolean variables, for which we give names like if_1, if_2, if_n etc. These variables (also called predicate variables) have only one of two possible values (0 or 1), representing if a particular condition is true (corresponds to value 1) or false (corresponds to value 0).

The basic building blocks of this system[§] consist of four formulae given in 1 to 4 below. In the following, X and a are variables with same units, e.g. inch.

```
1.     IF   X < a  THEN               if_n = sign( a – X )
             if_n = 1
       ELSE
             if_n = 0


2.     IF   X <= a  THEN              if_n = 1 - sign( X - a )
             if_n = 1
       ELSE
             if_n = 0


3.     IF   X > a  THEN               if_n = sign( X - a )
             if_n = 1
       ELSE
             if_n = 0


4.     IF   X >= a  THEN              if_n = 1 - sign( a – X )
             if_n = 1
       ELSE
             if_n = 0
```

We can use these building blocks to form compound logical statements.

[§] Although some of these functions can be obtained from the others, it is helpful to treat them as “basic” building blocks.

5. Logical AND (if more than two conditions are "connected" by AND's you can extend the same idea)

```
IF if_1 is true AND if_2 is true THEN     if_n = if_1 * if_2
      if_n = 1
ELSE
      If_n = 0
```

6. Logical OR (if more than two conditions are "connected" by OR's you can extend the same idea)

```
IF if_1 is true OR if_2 is true THEN   if_n = sign( if_1 + if_2 )
      if_n = 1
ELSE
      If_n = 0
```

7. Logical NOT (NOT true ≡ false)

```
IF if_1 is false THEN         if_n = 1 - if_1
      if_n = 1
ELSE
      If_n = 0
```

8. Exact Equality

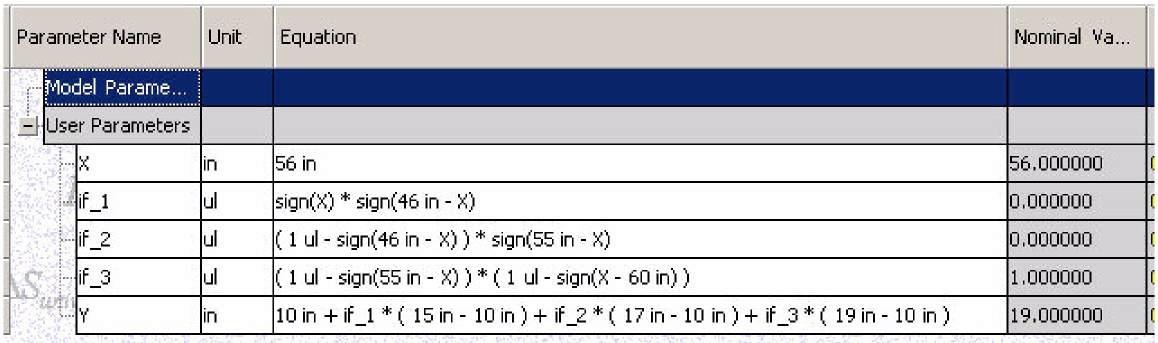
```
IF   X = a  THEN              if_n = (1 - sign( X - a )) *
      if_n = 1                            (1 - sign( a - X ))
ELSE
      if_n = 0
```

9. Equality with tolerance

```
IF   X = a ± tol THEN         if_n = (1 - sign( X - a - tol )) *
      if_n = 1                          (1 - sign( a - tol - X ))
ELSE
      if_n = 0
```

Now we can give the answer for the example shown at the beginning. For this, it is helpful to define three "if variables."

```
if_1 = sign(X) * sign(46 in - X)
if_2 = (1 ul - sign(46 in - X)) * sign(X - 55 in)
if_3 = (1 ul - sign(55 in - X)) * (1 ul - sign(X - 60 in))
Y    = 10 in + if_1 * (15 in - 10 in) + if_2 * (17 in - 10 in)
             + if_3 * (19 in - 10 in)
```



| Parameter Name | Unit | Equation | Nominal Va... |
|---|---|---|---|
| Model Parame... | | | |
| User Parameters | | | |
| X | in | 56 in | 56.000000 |
| if_1 | ul | sign(X) * sign(46 in - X) | 0.000000 |
| if_2 | ul | ( 1 ul - sign(46 in - X) ) * sign(55 in - X) | 0.000000 |
| if_3 | ul | ( 1 ul - sign(55 in - X) ) * ( 1 ul - sign(X - 60 in) ) | 1.000000 |
| Y | in | 10 in + if_1 * ( 15 in - 10 in ) + if_2 * ( 17 in - 10 in ) + if_3 * ( 19 in - 10 in ) | 19.000000 |