Function to test:

```
float Fido(float a, float b){
// compute delta(a-b)
print(a,b,a-b);
if (a == b )
        return 0; //target statement !
else
        return (a-b)/b;

}
```

we can change a or b by small values say 1 for simplicity, but it can be 0.001 or a random number between in [0,1], or any value you like. Assume we randomly select initial value of a=10 and b = 5.

If we consider close to a (and b) a+smallValue or a-smallValue for the smallValue =1 we have a+1 and a-1. Now to enter my target (the return 0) we must have a==b. In order to reach the goal, we can consider how different are a and b. This is to say compute a-b just before the if:

a	b	Delta (a-b)
10	5	5
10+1	5	6
10-1	5	4– better value
9+1	5	6
9	5+1	3 – better value

At each iteration we can choose between a+/- 1 or b+/- 1. Think   (a,b) are the coordinates of a point we move along the axes at step 1. Of course we can move also along any angle, for example I can move from (a,b) to (a+0,5, b-2.7) or any value I wish. The key is we only accept the new point if it improves current solution. Keep in mind our focus is to execute the target statement.

Can we do better, what is the lesson here ?

The Triangle Nasty Problem

Assume we get the 3 sides of a triangle and assume triange is initialize to zero:

```
1. if (side1 == side2) {
           triang = triang + 1;
}

2. if (side2 == side3) {
             triang = triang + 2;
}

3. if (side1 == side3) {
              triang = triang + 3;
}


4. if (triang == 1 && side1 + side2 > side3)
                            ret = ISOCELES;
}
```

To enter in 4 either we executed triang = triang + 1; or we will never enter 4! We have a data dependency BUT data dependencies are not modelled by simple test generation strategies!