Secant Method

Write the **R** function `secant.r` that calculates the root of the user-defined function f on a closed interval $[A, B]$ using the Secant Method.

Inputs:

1. A - one approximation of the root of the function f
2. B - a second approximation of the root of f
3. t - a user-defined tolerance

Output: C - a close approximation of a root of the function f

The function works by updating A and B each time through a *while* loop. The exiting criteria for the loop will be the same as the Bisection Method. Each time through the loop, the function calculates a value C such that C is the root of the line that passes through the points $(A, f(A))$ and $(B, f(B))$. The value of C is calculated using the formula

$$C = A - f(A)\frac{B - A}{f(B) - f(A)}.$$

Then, the new value of A becomes the old value of B, and the new value of B becomes the newly-calculated value C. The function repeats this process `while` the **absolute difference** between A and B is greater than the user-defined tolerance t:

$$|B - A|,$$

and the **relative difference** between A and B is greater than the user-defined tolerance t:

$$\frac{2\,|B - A|}{|A| + |B|} > t.$$

After the program has exited the `while` loop, `return` C.

Good luck. Here are several examples. For the first four examples, $f(x) = x^2 - e^x$. For the next five examples, $f(x) = \sin(x) - 2\cos(2x)$. And, for the last two examples,

$$f(x) = \begin{cases} -2x - 6, & x < -4 \\ x + 6, & -4 \le x < 0 \\ 6 - x^3, & 0 \le x \end{cases}$$

```
> f <- function(x){
+   y <- x^2 - exp(x)
+   return(y)
+ }
```

```
> secant(-2,2,0.000001)
[1] -0.7034674
> secant(2,-2,0.000001)
[1] -0.7034674
> secant(0,10,0.000001)
[1] -0.7034674
> secant(-10,-5,0.000001)
[1] -0.7034674
> f <- function(x){
+   y <- sin(x) - 2*cos(2*x)
+   return(y)
+ }
> secant(pi/2,-pi/2,0.000001)
[1] -2.138626
> secant(-pi/2,pi/2,0.000001)
[1] -1.002967
> secant(0,pi/2,0.000001)
[1] 0.6348669
> secant(pi/2,0,0.000001)
[1] 0.6348669
> secant(-10*pi,10*pi,0.000001)
[1] -2.232836e+23
> f <- function(x){
+   if (x < -4){
+     y <- -2*x - 6
+   } else if (x <= 0){
+     y <- x + 6
+   } else {
+     y <- 6 - x^3
+   }
+   return(y)
+ }
> secant(0,3,0.000001)
[1] 1.817121
> secant(3,0,0.000001)
[1] 1.817121
> secant(-5,-1,0.000001)
[1] -6.000003
```

You might wonder whether the algorithm worked on the last example.