Debug()

The function under test

For this explanation we will use the following simple function:

```
test = function(X)
{
  j = 0
  for (i in 1:X){
    j = j + i
  }
  j = j+1
  j
}
```

The function takes a single number as an argument and computes the sum of all numbers from 1 up to,and including, the argument. At the end 1 is added before the result is returned.

As an example;

```
> test(3)
[1] 7
```

The result is 7 because the functions loops from 1 till 3 and then cumulates each number and adds 1 , so 1 + 2 + 3 = 6 + 1 = 7

Setting breaks

We could start the debugger with;

```
> debug(test)
> test(3)
debugging in: test(3)
debug at #2: {
    j = 0
    for (i in 1:X) {
        j = j + i
    }
    browser()
    j = j + 1
    j
}
Browse[2]>

...
```

```
> undebug(test)
```

By pressing <enter> we will execute each line until the function returns. Debugging line per line is called single-stepping. The problem is what to do when have called the function for `x=10000000` and we are only interested in the line where `j=j+1`. We could single step for each line of the for loop for a million times, or we could use the following two techniques.

Call ''browser()''

We can modify our function by adding a call to `browser()` like this;

```
test = function(X)
{
  j = 0
  for (i in 1:X){
    j = j + i
  }
  browser()
  j = j+1
  j
}
```

In this case we do not need to call `debug(test)` because the `browser()` function will call the debugging environment for us. This way we can execute the function normally until the line with `browser()` is encountered. Do not forget to rerun the function after changing it or else the `browser()` command, and other added commands, will not get executed. For example;

```
> test = function(X)
+ {
+   j = 0
+   for (i in 1:X){
+     j = j + i
+   }
+   browser()
+   j = j+1
+   j
+ }
>
>
> test(1000000)
Called from: test(1e+06)
Browse[1]> j
[1] 500000500000
Browse[1]> i
[1] 1000000
Browse[1]> c
[1] 500000500001
```

Here we see that after running the function we automatically, without running `debug()`, get into debugging mode. Furthermore, after requesting the variable `i` we see it has a value of 1000000 so we have passed the for loop. If we would have another for loop we could just press `c` to continue until we meet the next `browser()` call.

Setting breakpoints

You should be able to stop the program at a certain point without calling the `browser()` command. Unfortunately I can't get it to work, maybe my version of R is different and it does work for you.

The breakpoints, places where the execution needs to stop, can be set by calling the `setBreakpoint()` function. For it to work you need to specify the filename of the script that you are trying to debug and the line number at which you would like to stop execution. Don't forget to save your file before executing this command or you might stop at the wrong line. As an example, where `a.r` is my file name;

```
> setBreakpoint("a.r#1")
No source refs found.
```

This should stop the execution when it reaches the specific line number, but I get the error `No source refs found.`

Conditional breakpoints

Besides just calling `browser()` we can also call it conditionally. We could be interested for example in the execution of just the 90000th iteration, or we could be interested in just that `NA` somewhere in our data files. Whatever the reason we can stop the executing simply like we did earlier but now combined with an `if` statement.

```
> test = function(X)
+ {
+   j = 0
+   for (i in 1:X){
+     if (i==90000) browser()
+     j = j + i
+   }
+   j = j+1
+   j
+ }
>
>
> test(1000000)
Called from: test(1e+06)
Browse[1]> i
[1] 90000
Browse[1]> c
[1] 500000500001
>
```