



MACQUARIE
University

Faculty of Science and Engineering

COMP125 Fundamentals of Computer Science
Workshop Week 9

Learning outcomes

By the end of this session, you will have learnt about recursions.

1. Recursion trace

Consider the following recursive function definition,

```
1 int foo(int a) {  
2     if(a == 2)  
3         return 2;  
4     return a + foo(a / 2);  
5 }
```

What is the value of variable `result` if the function call is,

```
1 int result = foo(16);
```

Solution:

```
foo(16) = 16 + foo(8)  
foo(8) = 8 + foo(4)  
foo(4) = 4 + foo(2)  
foo(2) = 2  
Therefore,  
foo(4) = 4 + 2 = 6  
foo(8) = 8 + 6 = 14  
foo(16) = 16 + 14 = 30
```

2. Debugging recursive functions

The following function attempts to compute the factorial of integer n . What is wrong with the function?

```
1 int factorial(int n) {  
2     return n * factorial(n - 1);  
3     if(n == 0)  
4         return 0;  
5 }
```

Solution:

- a. Termination conditions should be BEFORE recursive call.
- b. Termination condition is incorrect. Should return 1 if $n \leq 1$ otherwise factorial will become 0.

```

1 int factorial(int n) {
2     if(n <= 1)
3         return 1;
4     return n * factorial(n - 1);
5 }

```

3. Debugging recursive functions

Give an example of a value, that, if passed to the function `foo` from the previous question, calls itself indefinitely.

Solution: `foo(24) = 24 + foo(12)` `foo(12) = 12 + foo(6)` `foo(6) = 6 + foo(3)` `foo(3) = 3 + foo(1)` `foo(1) = 1 + foo(0)` `foo(0) = 0 + foo(0) ...`

4. Some more recursive trace

Consider the following recursive function definition,

```

1 int foo(int a) {
2     if(a <= 0)
3         return 0;
4     if(a % 2 == 0)
5         return foo(a/2);
6     else
7         return 1 + foo(a/2);
8 }

```

What is the value of variable `result` if the function call is,

```

1 int result = foo(59);

```

Solution:

```

foo(59) = 1 + foo(29)
foo(29) = 1 + foo(14)
foo(14) = foo(7)
foo(7) = 1 + foo(3)
foo(3) = 1 + foo(1)
foo(1) = 1 + foo(0)
foo(0) = 0
Therefore, foo(59) = 5

```

5. Writing a recursive function

Write a recursive function, that when passed an integer, returns the number of even digits in that integer. Return 0 if the integer is 0.

Solution:

```
1 int nEvenDigits(int n) {
2     if(n == 0)
3         return 0;
4     if(n%2 == 0)
5         return 1 + nEvenDigits(n/10);
6     else
7         return nEvenDigits(n/10);
8 }
```

6. Writing a recursive function

Write a recursive function, that when passed an integer n , return the sum of squares of the first n positive integers ($1 + 2 + \dots + n$).

Solution:

```
1 int sumSquares(int n) {
2     if(n <= 0)
3         return 0;
4     return n*n + sumSquares(n - 1);
5 }
```

7. Writing a recursive function dealing with text

Write a recursive function, that when passed a String, returns the number of digits in the String.

Solution:

```
1 int nDigits(String s) {
2     if(s == null || s.length() == 0)
3         return 0;
4     if(s.charAt(0) >= '0' && s.charAt(0) <= '9')
5         return 1 + nDigits(s.substring(1));
6     return nDigits(s.substring(1));
7 }
```

8. Counting recursive function calls

How many calls are made to gcd if the original call is gcd(30, 72)?

```
1 int gcd(int a, int b) {
2     if(a < b)
3         return gcd(b, a);
4     if(b == 0)
5         return a;
6     return gcd(b, a%b);
7 }
```

Solution:

```
gcd(30, 72) = gd(72, 30)
gcd(72, 30) = gcd(30, 12)
gcd(30, 12) = gcd(12, 6)
gcd(12, 6) = gcd(6, 0)
gcd(6, 0) = 6
```

a total of 5 function calls

9. (Tracing slightly more complex recursive functions)

Consider the definition of the following recursive function,

```
1 public static void displayBrackets(int n) {
2     if(n == 0)
3         return;
4     System.out.print("(");
5     for(int i=0; i < n - 2; i++) {
6         displayBrackets(n - 1);
7         System.out.print(",_");
8     }
9     displayBrackets(n - 1);
10    System.out.print(")");
11 }
```

What is the output of the following statement?

```
1 displayBrackets(3);
```

10. (Assessed task) Defining recursive functions

I have made up a sequence called a *tribonacci* sequence. The first three numbers of this sequence are 1, 2 and 3, and every subsequent number in this sequence is the sum of the previous **three** numbers. Thus, the sequence is 1, 2, 3, 6, 11, 20, 37, 68, Write a function to compute the n^{th} *tribonacci* number. Assuming the 1st number is 1.

Solution:

```
1 int tribonacci(int n) {
2     if(n < 4)
3         return n;
4     return tribonacci(n - 1) + tribonacci(n - 2) + tribonacci(n - 3);
5 }
```

11. (Assessed task) Counting recursive function calls

How many calls are made to `tribonacci` if the original call is `tribonacci(5)`?

Solution:

```
t(5) = t(4) + t(3) + t(2)
t(4) = t(3) + t(2) + t(1)
t(3) called twice returns 3 each time
t(2) called twice returns 2 each time
t(1) called once returns 1
```

So termination level calls are 5, and in addition `t(4)` and `t(5)` once each. So a total of 7 function calls

12. (Voluntary Assessed task) Writing a recursive function

Write a recursive function that displays an hour-glass pattern. For example, it displays the following pattern for $n = 5$.

```

* * * * *
  * * * * *
    * * * * *
      * * * *
        * * *
          *
            *
          * * *
        * * * * *
      * * * * * *
    * * * * * * *
  * * * * * * * *
* * * * * * * * *

```

And it displays the following pattern for $n = 7$.

```
*****  
*****  
      *****  
        *****  
          *****  
            *****  
              ****  
                ***  
                  *  
                    *  
                      ***  
                        *****  
                          *****  
                            *****  
                              *****  
                                *****  
                                  *****  
                                    *****
```