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,

```
int foo(int a) {
        if(a == 2)
                return 2;
        return a + foo(a / 2);
}
```

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

```
        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?

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

   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.

```
int factorial(int n) {
        if(n <= 1)
                return 1;
        return n * factorial(n - 1);
}
```

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,

```
int foo(int a) {
        if(a <= 0)
                return 0;
        if(a % 2 == 0)
                return foo(a/2);
        else
                return 1 + foo(a/2);
}
```

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

```
        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:

```
int nEvenDigits(int n) {
        if(n == 0)
                return 0;
        if(n%2 == 0)
                return 1 + nEvenDigits(n/10);
        else
                return nEvenDigits(n/10);
}
```

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 $(1+2+...+n)$.

Solution:

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

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:

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

8. **Counting recursive function calls**

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

```
int gcd(int a, int b) {
        if(a < b)
                return gcd(b, a);
        if(b == 0)
                return a;
        return gcd(b, a%b);
```

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 1^{st} 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) + (t1)
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 c
```

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 $n = 7$.

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