Department of Computing

COMP125 Fundamentals of Computer Science
Workshop - Recursion

Learning outcomes

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

1. **Recursion trace**

   Consider the following recursive function definition,

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

   What is the value of variable `status` if the method call is,

```
        int status = foo(8);
```

   Provide a trace of the code.

2. **Debugging recursive methods**

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

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

   Give an example of a value, that, if passed to the method `foo`, results in `StackOverflowError` (method calls itself indefinitely).

3. **Some more recursive trace**

Consider the following recursive method 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 method call is,

```
        int result = foo(59);
```

4. **Writing a recursive method**

   a. Write a recursive method, that when passed an integer, returns the number of even digits in that integer. Return 0 if the integer is 0. Some input-output mappings:

   ```
   4729  -> 1
   13579 -> 0
   284   -> 3
   ```

   Tip: `n%10` gives the last digit of integer `n` and `n/10` gives the the number with last digit removed.

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

5. **Writing a recursive method dealing with text**

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

Tip: expression to check if a character `ch` is a digit is `(ch >= '0' && ch <= '9')`

6. **Counting recursive method calls**

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

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

7. **(Tracing slightly more complex recursive methods)**

   Consider the definition of the following recursive method,

```
public static void displayBrackets(int n) {
        if(n == 0)
                return;
        System.out.print("{");
        for(int i=0; i < n - 2; i++) {
                displayBrackets(n - 1);
                System.out.print(", ");
        }
        displayBrackets(n - 1);
        System.out.print("}");
}
```

   What is the output of the following statement?

```
displayBrackets(3);
```

8. **(Time permitting) Defining recursive methods**

   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 $1, 2, 3, 6, 11, 20, 37, 68, \ldots$. Write a method to compute the n^{th} *tribonacci* number. Assuming the 1^{st} number is 1.

9. **(Time-permitting otherwise take-home task) Counting recursive method calls**

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

10. **(Time-permitting otherwise take-home task) Writing a recursive method**

Write a recursive method that displays an hour-glass pattern. For example, it displays the following pattern for $n = 6$. You MAY add more parameters to the method if required.

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

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

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

11. **CHALLENGING (Time-permitting otherwise take-home task)**

Write a method that when passed a String containing letters of the English alphabet (you may assume that each letter occurs only once), returns an array of Strings containing all combinations of the letters from the input String.

```
public static String[] getCombinations(String s)
```

For example, if `s = "abcd"`, the method should return the String

```
{
 "abcd", "abdc", "acbd", "acdb", "adbc", "adcb",
 "bacd", "badc", "bcad", "bcda", "bdac", "bdca",
 "cabd", "cadb", "cbad", "cbda", "cdab", "cdba",
 "dabc", "dacb", "dbac", "dbca", "dcab", "dcba"
}
```