COMP1917: 16 Recursion

Sim Mautner

s.mautner@unsw.edu.au

October 9, 2016

Recursion

- A recursive function is a function which calls itself.
- It does this with a smaller version of the problem, to get the task partially done.
- Recursion can do anything a Loop can. And vice versa.
- So why do we use recursion? Why not just use loops all the time?
  - Some situations (which are specifically covered in COMP1927) are significantly easier to solve using recursion than by using loops.
  - For now we will just be looking at producing recursive solutions to simple loops problems.

How Recursion Works

- Marking exams analogy

The Structure of a Recursive Function

- A base condition (when the function should cease calling itself).
- One or more recursive conditions (when the function should call itself).

```
void aFunc(/* parameters */) {
    if(/* base case */) {
        return; // or return returnValue; for non-void funct
    } else {
        aFunc(/* altered parameters */);
        return;
    }
}
```

Side Note: Style Guide for Recursive Functions

- These can be written without multiple return statements.
- For teaching purposes I use multiple return statements. (I feel students are more likely to understand this approach, learn quicker etc.)
- Marks will not be deducted for using multiple return statements in recursive functions.

Exercises

- Ex 1: Write a recursive function which takes in an integer and prints a line of that many asterisks.
  1. Re-write this function using only one return statement.
  2. Compare this function to the equivalent function written using a loop.
- Ex 2: Write a recursive function which given a number, prints the numbers from that number down to 0.
- Ex 3: Write a recursive function which prints out a string, letter by letter.
- Ex 4: Write a recursive function which prints out a string *backwards*, letter by letter.
- Ex 5: Write a recursive function which given a number, prints out that many fibonacci numbers.