PAL Consolidation 4 *University of Hertfordshire – Week 6 – November 2015 –*

1 Aim & Objectives

- Re-cap previous sessions and answers
- Work through lecture related exercises

2 Introduction

In this session we'll be looking at procedural programming again, but in even more detail than the previous session. We will also lightly touch on recursive functions but this will be covered better at a later date.

3 Exercise - Pseudo Code

Writing pseudo code for functions is relatively easy and follows the following format:

```
PROCEDURE hello()
   DISPLAY "Hello World!"
ENDPROCEDURE

PROCEDURE functionName()
   hello()
ENDPROCEDURE

functionName()
```

Now, using what you've been show here try to write the pseudo code for a program that calls a function called '`A()`', which calls a function '`B()`', which then calls a function '`C()`'. In the '`C()`' function, get the function to write "*Three layers down!*".

It's generally good practice to declare functions before you use them in most languages, it's sometimes completely required - so it's a good habit to get into! A scripting language like '`BASH`' requires you declare all functions before you use them.

```
PROCEDURE C()
   DISPLAY "Three layers down"
ENDPROCEDURE

PROCEDURE B()
   C()
ENDPROCEDURE

PROCEDURE A()
   B()
ENDPROCEDURE

A()
```

4 Exercise - Python Code

Now write the Python code you described in the previous exercise. This should be relatively easy by now but if you need hints please refer to the previous week for clues.

It's getting towards the point where your programs will become complex enough for you to start forgetting how they work. Make sure you start commenting your code correctly, even in little code snippets like these.

```
# Simple Example of Python Functions
#
# @author D.Barry
# @version 1

# C()
#
# This function displays a message before
exiting.
def C():
   print("Three layers down")
   return

# B()
#
# This function calls function C().
def B():
   C()
   return

# A()
#
# This function calls function A().
def A():
   B()
   return

# Call the first function
A()
```

Note that a '`Java`' style comment has been used. Haskell does have it's own way of commenting code but in this example '`Java`' style has been opted for readability.

5 Exercise - Simple Recursion

For this next session you'll have to do a little self learning. This should help you understand recursion better.

- What is recursion? What is the definition of recursion?
- What does recursion need in order to work?
- What uses are there for recursion?
- Are there tasks that can *only* be done with recursion?

Now try to write your own recursive function that iterates a number of times.

6 Exercise - Stretch and Reach

Try these exercises if you really want to push yourself. There is almost zero chance they will appear on the course at all but act as interesting discussion points.

- Research "*Trotter's Algorithm*" - what is it?
- What kinds of programs would require you to use this program?
- Why is this program easier to implement recursively?
- Discuss why some functions simply can't be changed from recursive to iterative (loops).
- Research problem spaces and the $N = NP$ problem.

7 Resources & Further Reading

'*http://homepages.herts.ac.uk/~db12aba/*' – All content from these sessions updated weekly.

'*http://code.org/*' – A good resource testing your programming skills.

'*http://stackoverflow.com/*' – Highly recommended online help for programmers (NOTE: Employers are interested to know whether you're an active member of this site!).

'*http://www.draw.io*' – A very good, free online drawing tool that exports to many formats, including '`XML`' and '`JPG`'.