EE101 C programming and SW engineering 1
Lab Practice 6 – Operators and Functions

Use your preferred compiler to investigate the programming exercises below. This laboratory concerns an investigation of operations including the increment and decrement operators ++ and -- as well as some accumulators such as +=, *=, etc. You will also design, write and document simple modular programs that use functions.

Operations

Exercise 1

Study the following code and write down what you think is printed. Also, provide a simple justification for each answer.

```
#include <stdio.h>

int main(void){
int a, b=0, c=0;
a = ++b + ++c;
printf("%d %d %d\n", a , b ,c);
a = ++b + c++;
printf("%d %d %d\n", a , b ,c);
a = b-- + --c;
printf("%d %d %d\n", a , b ,c);
}
```

Here are some additional assignment operators

+= -= *= /=

which can be used to simplify your statements. The general form of a statement involving these operators is:

variable op= expression;

where variable can be any numerical variable you have defined in your program and op is an operator such as + - * / etc.

The previous statement is equivalent with:

variable = variable op (expression);

Note the parenthesis surrounding expression.

The following table illustrates how assignment expressions are evaluated.

Declarations and Initialisations			
int i=1, j=2, m=3, k=4;			
Expression	Parenthesised	Equivalent	Value
i += j+k	i += (j+k)	i = i + (j+k)	7
j *= k=m+5	j *= (k=(m+5))	j = j * (k=(m+5))	16

Exercise 2

Consider the following code:

```
int a = 1, b = 2, c = 3;
a += b += c += 7;
```

Write an equivalent statement that it is fully parenthesised. What are the final values of the variables a, b and c?

Exercise 3

Find and write down the value of quack after each line:

```
int quack = 2;
quack += 5;
quack *= 10;
quack -= 6;
quack /= 8;
```

Exercise 4

Write a program that asks the user to input an integer and then prints all the integers from (and including) that value up to (and including) a value larger by 10. That is, if the input is 5 the output runs from 5 to 15. Make sure you separate each output value by a space, comma or a new line.

Use a **while** loop to keep doing all of the above, until the user inputs a character.

Basic Functions

Exercise 5

Write a function called **mul2** that takes as its argument an **int**eger and that returns an **int**eger to say if the input argument is a multiple of 2.

Write a second function **mul3** that takes as argument an **int**eger and that returns an **int**eger to say if the input argument is a multiple of 3.

Use these two functions to write a program that reads in an integer and that indicates if the input integer is odd, not a multiple of 3 and not a multiple of 6.

Hint: the % operator returns the remainder of a division

Exercise 6

The Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, begins with 0 and 1 and has the property that each subsequent Fibonacci number is the sum of the previous two Fibonacci numbers.

Write a recursive function fibonacci(i) that calculates the ith Fibonacci number and then use it to write a program that reads in an **int**eger number n and prints out the nth Fibonacci number.

Exercise 7

A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5, and 6 spots.

Craps is a gambling game played in casinos using dice. The rules are as follows:

- The player rolls the dice and the sum of the spots on the two upward faces is calculated.
- If the sum is 7 or 11 on the first throw, the player wins.
- If the sum is 2, 3, or 12 on the first throw (called “craps”), the player loses (casino wins).
- If the sum is 4, 5, 6, 8, 9, or 10 on the first throw, then that sum becomes the player’s “point”.
- To win, you must continue rolling the dice until you make your “point” again.
- The player loses by rolling a 7 before making the “point”.

Using the C random number generator, write a program that simulates this game.

Hint: You might need to include the header file time.h in the program which contains definitions of functions to get and manipulate date and time information