Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming (2021/22)
Laboratory Session Week_3

Learning Objectives

- To use C functions (e.g. **`rand()`**[1], **`scanf()`**) and write simple user-defined C functions.
- To apply appropriate pre-processor directives when solving problems.
- To use multiple-dimensional arrays, where appropriate.

1. Exercises

You should attempt the exercises below by using only the C constructs that you learnt up to teaching week 3, and:

1. Write pseudo code to describe the required algorithm to solve the exercise (or draw up a flowchart), before writing and testing the actual code.
2. Add comments to your code.
3. Make your code neat, by using indentation and parenthesis (where appropriate).
4. Give meaningful names to functions and variables.

Exercise 1

First part

Write a program that generates a random number between two values, which are set in the program with pre-processor directives. The program should then ask the user to guess what number the computer 'thought' of. If the user guessed the number correctly, then the program should print a **`"Well done!"`** message and exit. Otherwise, the program should print the message **`"Wrong guess; better luck next time!"`** and exit. Save your program to a file called **`guessingNumber.c`**.

Second part

You should then extend the program you wrote, such that your new program gives the user a hint if the number is guessed incorrectly, i.e.
`"The correct number is bigger"` or **`"The correct number is smaller"`**. Save your new program to a file called **`guessingNumber_better.c`**.

[1] Do not use **`srand()`**; we did not cover this in class, and it is not necessary in order to solve the exercises.

Exercise 2

Write a program that implements a simple calculator with the four basic operations for integer values: addition, subtraction, multiplication, and division. You should use a function-like macro to define each operation. The program should read the following data from the input stream (in this case, the keyboard): two integer values, and a character value to select the chosen operation. The program then applies the adequate function-like macro to determine and display the resulting value. Save your program to a file called **`simpleCalculator.c`**.

Hint: Use **`scanf()`** to read in the 2 numbers and the character.

Exercise 3

Modify the program you wrote in exercise 4 of the **Lab 1** sheet, such that the functions for the three different shapes (i.e. the two triangles and the square) are now defined in a separate file called **`shapes.c`** and their prototypes in **`shapes.h`**. Each function should have an integer parameter called **`size`**. In a separate file called **`main.c`**, the **`main()`** function will invoke the 3 shape functions. Your program (i.e. file **`main.c`**) should also define a global character variable to set the character (e.g. '*') with which shapes will be filled. Once you have written the source code, you should be able to generate the executable code by typing the following commands:

```
$ gcc –c shapes.c
$ gcc main.c shapes.o –o printShapes
```

Figure 1

Exercise 4

Write a program that uses a two-dimensional array (defined in **Figure 2**) containing the marks for 3 students in 4 modules, to determine and display the average mark for each student. Your program should include a user-defined C function – called **`average()`** – for calculating the average mark of each student. **Figure 3** shows the expected program output. Save your program to a file called **`averageGrader.c`**.

```
double std_grades[3][4] = {{7.7, 6.8, 8.6, 7.3},
                           {9.6, 8.7, 8.9, 7.8},
                           {7.0, 9.0, 8.6, 8.1}};
```

Figure 2

```
$ ./averageGrader

Grades for student 0 are: 7.7 6.8 8.6 7.3
Student average = 7.60

Grades for student 1 are: 9.6 8.7 8.9 7.8
Student average = 8.75

Grades for student 2 are: 7.0 9.0 8.6 8.1
Student average = 8.18
```

Figure 3

ECS501U – END of LAB 3