Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming (2021/22)
Laboratory Session Week_4

Learning Objectives

- To use multiple-dimensional arrays and write C programs which use the notion of pointers.
- To apply appropriate data structures (e.g. **`struct`**) when solving problems.
- To apply basic string concepts/functions (e.g. **`strlen()`**) when solving problems.

1. Exercises

You should attempt the exercises below by using only the C constructs that you learnt up to teaching week 4, 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

Declare a structure called **`student`** which contains student details: name, number and an array of marks. Create an array of 5 students defined by the **`student`** structure and write a program that uses this to print out the name and average mark for each student. Save your program to a file called **`classAverageGrader.c`**.

Exercise 2

Read the data in **Figure 1** into an array, which represents all the QMUL's scores in a sports' league. The program should then print to the output stream (in this case, the screen) a summary of QMUL's performance in a league table format, as shown in **Figure 2**. Save your program to a file called **`uniSports.c`**.

```
QMUL 3 Kings 0
Oxford 5 QMUL 4
QMUL 2 Imperial 5
Cambridge 1 QMUL 2
QMUL 5 Essex 3
Brunel 2 QMUL 2
QMUL 6 LSE 4
UCL 4 QMUL 3
QMUL 1 Kent 3
Surrey 3 QMUL 3
```

Figure 1 – `qmulResults.txt`

```
$ ./uniSports < qmulResults.txt

     Home   Away
     W D L  W D L  GF GA GD  PTs
QMUL 3 0 2  1 2 2  31 30  1   14
```

Figure 2

Notes:

- The example shown in **Figure 2** uses a football points' scoring system, i.e. 3 points for a win (**W**), 1 point for a draw (**D**) and no points for a game lost (**L**).
- **GF** (Goals For) refers to the number of goals a team has scored, while **GA** (Goals Against) refers to the number of goals scored against a team.
- **GD** (Goals Difference) refers to the difference between **GF** and **GA**, while **PTs** (Points) is the total number of points earned by the team from all the games played (i.e. taking into account those won, drawn and lost).
- In **Figure 2**, we have PTs = 4*W + 2*D + 4*L = 12 + 2 + 0 = 14.

Exercise 3

Write a C function called **`stringDelete()`**, such that it deletes characters inside of a string. In particular, the **`stringDelete()`** function will shift all characters that are to the right of the chosen character, by one position to the left.

Write a small driver program to test your **`stringDelete()`** function and save it to a file called **`test_stringDelete.c`**. **Figure 3** shows the expected program output when the user enters the word **`lettter`** and position **`3`** on the command line:

```
$ ./test_stringDelete
Type in the word and press Enter: lettter
Type in the position of the character to be deleted: 3

Original word = lettter
Modified word = letter
```

Figure 3

Hint: Think carefully about the data type of the parameters of your function, as well as which string related function(s) might be useful to apply.

Exercise 4

Write a C function called **`stringInsert()`** that inserts a character in any position of a string entered by the user. The function prototype should be as follows:

```
void stringInsert(char str[], char c, int position);
```

Write a small driver program to test your **`stringInsert()`** function and save it to a file called **`test_stringInsert.c`**.

ECS501U – END of LAB 4