Queen Mary
University of London

School of Electronic Engineering and Computer Science

ECS501U – C Programming (2017/18)
Laboratory Session Week_2

Learning Objectives

- To use C functions (e.g. **`getchar()`**) and write simple user-defined C functions.
- To define and use arrays.

1. Exercises

You should attempt the exercises below by using only the C constructs that you learnt up to teaching week 2, 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

Write a program that reads a set of characters from the input stream (in this case, the keyboard) and determines how many of those characters are spaces (" ") and how many are new lines ("\n"). The program should read characters until the user presses [ctrl-D] (to symbolise the end of file, represented with the constant **EOF**) and it should print the total number of spaces and new lines found, to the output stream (in this case, the screen). Save your program to a file called **`count_spacesAndNewLines.c`**.

To test this program with large amounts of text, we can use ***redirection***, without having to change the program. This means that instead of reading characters from the keyboard, your program can read text from a file. To do this, simply run the program as indicated below, with text file **`poem.txt`**, which you can download from the **C Programming Labs** topic on the ECS501U course website.

```
$ ./count_spacesAndNewLines < poem.txt
```

Figure 1

Note: You can use other text files to do your testing.

Exercise 2

Modify the program you wrote in exercise 1, such that it counts the number of capital letters in the set of characters read from the input stream. The program should print the total number of capital letters found, to the output stream (in this case, the screen). Save your program to a file called **`count_capLetters.c`**.

Hint: Consider the range of ASCII encoding values which represent the letters of the alphabet.

Exercise 3

Write a program that reads a fixed length word (e.g. 5 characters long) from the input stream (in this case, the keyboard), and determines whether the word is a *palindrome* (e.g. the word "level"). Your program should, accordingly, simply print one of the messages to the output stream:

`"This word is a palindrome."`
`"This is not a palindrome."`

Save your program to a file called **`palindromeChecher.c`**.

Hint: You may want to store the word read from the input stream in an array.

Exercise 4

Write a program that reads in 10 positive integers from the input stream and draws the corresponding vertical bar chart; below is an example of running the program. Save your program to a file called **`verticalBarChart.c`**.

```
$ ./verticalBarChart
Please enter 10 numbers:
6
5
2
0
3
1
4
3
3
0

*
* *
* *           *
* *       *   * * *
* * *     *   * * *
* * *     * * * * *
```

Figure 2

Hint: Use **`scanf()`** to read in the 10 numbers and an array to store them.
Note: If you manage to do this exercise, try modifying the program to also handle negative numbers.

ECS501U – END of Lab Week_2