CS264 Laboratory Session 2

Dr. Edgar Galván

3^{rd} Oct 2017

Deadline: All solutions to be submitted by 5pm Tuesday 10^{th} October 2017

1 Lab objectives

In this lab you will continue on from last week by solving (slightly) more complex problems in C++. In particular, you will be required to write programs that make use of the features of C++ that we covered since the last lab.

Learning Outcomes
By the end of this lab sheet you should be able to:

1. Write simple C++ programs that make use of (i) functions (including pass by reference arguments), (ii) arrays, (iii) c-strings, and (iv) pointers.

For each of the problems given below write a C++ program that provides a solution that problem. Name each solution as **exercise_1.cpp, exercise_2.cpp**, etc. Failure to do so can result in loss of marks. **You should include a comment at the top of each source file with your full name followed by your surname, the latter in capital letters (e.g., Edgar GALVAN) and student number (if you have one).**

2 Questions:

For each of the problems given below write a C++ program that provides a solution. Each box provides a filename to use (or in certain cases multiple filenames). Please ensure that you use those filenames.

Step 0.1: For this week's exercises you should create a sub-directory off of the top level `Labs` directory from last week called `Lab2`. Once you have created your first file within that directory you should add it to your repository. All your work for this assignment should go into that directory. Be sure to commit at least after each exercise with appropriate commit messages.

Remember:

- **Add source files to your git repository and commit changes regularly.**
- **All commits should be accompanied by messages that would allow a lecturer or demonstrator to understand the purpose of the commit.**
- **Comment your code.**

- **Use proper indentation for function and control structures.**

Exercise 1: Write a function, called `my_swap`, that takes two integers as input and swaps their values. To demonstrate that the function works you should write a program that instructs the user to input two numbers. The program should store those numbers in two separate variables. The program should then output the values stored in the two variables to the screen, swap the two numbers by calling `my_swap`, and then again print out the values stored in the two variables. You should save the source in a file called `exercise1.cpp`.

Exercise 2: The greatest common divisor (GCD) of two integers is the largest integer that evenly divides into each of the two integers. Write a function called `gcd` that returns the greatest common divisor of two integers. The program should contain a function called `gcd`, with appropriate parameters and return type, that provide all of the logic for computing the gcd of the two inputted numbers. You should save the source in a file called `exercise2.cpp`.

Exercise 3: Write a program that simulates the rolling of two dice. The program should call `rand` to roll the first die, and call `rand` again to roll the second die. The sum of the two values should then be calculated. Your program should roll the pair of dice a user specified number of times, keeping track of the number of times each possible *total value* occurs. To do this it should use an array of 11 items (since there are only 11 possible outcomes i.e. 2 to 12). At the end of the run the program should print out, in a tabulated format, the percentage of times each possible total value occurs. You should save the source in a file called `exercise3.cpp`.

Exercise 4: Write two functions with the following function prototypes:
`int my_string_len(char str[])`
`void my_string_cat(char dest[], char src[], int dest_size)`
Both functions should take *zero-terminated* strings as input. The first function should return the length of the string to the calling function. The second function should take a source and a destination string and the total size of the array pointed to by `dest`. It should then concatenate the source string onto the end of the destination string, **if and only if the destination string has the capacity to store both strings**. If the destination string does not have the capacity it should not alter either, print and error, and return to the calling function.

Note: your functions should not use library functions such as strlen or strcat.

Demonstrate the use of both the above functions by using them in a program in which you initialise two character arrays with two strings, print out their length, print out the combined length, and print out the combined string.

You should save the source in a file called `exercise4.cpp`.

Exercise 5: Write two functions with the following function prototypes:
`void print_string_array(char str[])`
`void print_string_ptr(char *str)`
Both of the above functions should print *zero-terminated* c-strings to the screen character by character *i.e.* by stepping through the string one character at a time and returning once a '\0' character is encountered. The first function should use the array indexing operator to achieve this, whilst the second should use a pointer to traverse the characters of the string.

Demonstrate the use both the above functions by using them in a program to print out a string which is pre-initialised in the main function

You should save the source in a file called `exercise5.cpp`.