EE208 Laboratory Session 3

Bryan Hennelly

13th February 2018

1 Lab objectives

In this lab you will continue on from last week by solving (slightly) more complex problems in C++ and yu can continue to use the online compiler for one more week. In particular, you will be required to write programs that make use of the features of C++ that we covered since the last lab, namely and arrays and pointers. D'ont forget to make full use of the online C++ tutorial if you are struggling with any of the concepts. You have one week to submit all of your solutions to this weeks lab on moodle; the assignments are due at 5pm Monday 19th Feb. This lab is worth 2% of your overall grade. Don't forget to comment your code for full marks. The demonstrators are here to help so don't be afraid to shout if you're having difficulties.

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, (ii) arrays, (iii) pointers, and (iv) dynamic memory allocation; you will have to peek ahead at the next lecture to see how to implement dynamic memory allocation.
2. Have a thorough understanding of the relationship between arrays and pointers.

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. If you wish yu can submit your code to moodle.

Remember:

- **Comment your code.**
- **Use proper indentation for function and control structure.**

Exercise 3.1: Write a program that creates an array of integers of of size 10; you should declare the array and set the 10 values at the same time using the chain brackets. The program must output the largest element in the array, and the index at which that element was found. You should save the source in a file called `exercise_3_1.cpp`

Exercise 3.2: Repeat exercise 1 but this time write a function to find the maximum value, called `find_max`, which takes in an array and returns both the largest element in the array, and the index at which that element was found. Demonstrate your function working in the main function. Hint: maybe you should return an array with two values. You should save the source in a file called `exercise_3_2.cpp`.

Exercise 3.3: Repeat exercise 2 but this time your program/function should work for an array of length N, where N is input by the user; the user should be asked to enter each value one at a time. You must use a pointer and dynamic allocation to create the array. You will also need to change your function to accept a pointer as input, rather than an array. You should save the source in a file called `exercise_3_3.cpp`.

Exercise 3.4: Write a function that receives as input two arrays of doubles of arbitrary length. The function should returns an array of doubles that is made up of a concatenation of the two input arrays. Demonstrate your function in action in your main function by printing the full contents of the new array. Hint: you will need to use dynamic memory allocation to create the new array and you wont actually return an array, rather you must return a pointer. You should save the source in a file called `exercise_3_4.cpp`.

Exercise 3.5: tricky! Write a program that asks the user to enter an integer, N and then use dynamic memory allocation to create an array of integers of size N. Next populate this array with random numbers between 1 and 1000; you should have learned how to do this last week. Next, write a function to sort the array in increasing order. You should display the result in your main function. Hint: maybe you should think about using the swap function that you wrote last week. You should save the source in a file called `exercise_3_5.cpp`.

Exercise 3.6: difficult! Write a program that creates two 4x4 arrays of doubles, which represent two 2D matrices. You can set the values when you declare the arrays using the chain brackets. Now write a function that takes in both 'matrices' and return the matrix product. Demonstrate your function in action in the main function; print the values of the result to screen with clear formatting. You should save the source in a file called `exercise_3_6.cpp`.