60-141 – Introduction to Programming II Winter, 2017

Lab #4: Pointers

(Due at the beginning of the next lab period)

Objective: Learn to use pointers.

1. Consider the following C program. Assume that memory addresses are expressed in decimal numbers and an integer takes 4 bytes. Also, assume **ids = &ids[0] = 2000**. What would be printed by the following program? Do NOT compile and execute this program as it will not provide the correct results – do the tracing of the program by hand and using only the assumptions stated.

```
#include <stdio.h>
int main()
 {
    int ids[3] = {100,200,300};
    int *salary, salary1, salary2, *salary3;
    salary1 = ids[0] * ids[1];
    salary = &ids[1] ;
    salary2 = *(ids+1)* *(ids+2);
    salary3 = ids+2;
    printf("*salary = %d\nsalary1 = %d\n", *salary, salary1);
    printf("salary2 = %d\nsalary3 = %p\n", salary2, salary3);
    return 0;
}
```

2. Write a documented function called **Largest** that finds and returns the **address of the largest element** in the array passed to it. (Assume an integer array of size 10). You must use **pointer arithmetic on the array name** instead of the array subscript notation to implement the function. (NOTE: This function will be included as part of step 4 below.)

```
int *Largest( int *array, int size );
```

3. Write a documented function called **Swap** that takes two integer **pointers** and exchanges the values of each. It returns void. Example: given two integers 'a = 2' and 'b = 4', after **Swap (&a, &b)** is called, 'a' will be 4 and 'b' will be 2. (NOTE: This function will be included as part of step 4 below.)

```
void Swap( int *x, int *y );
```

4. Write a complete, well documented C language program (called **Lab4.c**) to test both the functions **Largest** and **Swap**. In order to test your functions you will need to define and populate an array, within main, that you can pass to **Largest,** and also two initialized int variables to pass into **Swap**; do not forget to write printf() statements to show Before/After views of any data that will undergo changes. Ensure that your testing clearly demonstrate each function and its required result. There is no need for a menu, nor for the program logic to repeat itself (unless you choose to implement such logic).

Summary of the lab requirements: You must complete Part 1 above by hand, without using a computer or doing any programming. Parts 2-4 must be completed by writing a single C language program that will be compiled and executed to demonstrate the results. Remember to fully document your programs.

EVALUATION OF WORK AND ATTENDANCE: Total 5 marks.
You need to show to your lab instructor the work you have completed for this lab assignment, generally in the form of a working program. The marks you will receive for the lab are made of two parts, the programming part and lab attendance. Do not email your work to the Instructor or to any teaching assistant.

Lab Work Mark: You will be evaluated based on your solution for the problem assigned, using the following scheme: You must attend the lab and show your work in order to earn any of the following marks.

0 mark = No appreciable and relevant work done.
1 mark = Incomplete code / does not compile, with no, little or invalid documentation.
2 marks = Complete running program with no, little or invalid documentation.
3 marks = Incomplete code / does not compile, with suitable documentation.
4 marks = Complete running program with suitable documentation (minor errors may be accepted).

Attendance Mark: You will receive 1 mark for attendance during the lab period.

IMPORTANT:
ASK QUESTIONS IF YOU GET STUCK, BUT DO YOUR OWN CODING. SUBMITTING WORK COPIED FROM OTHER STUDENTS WILL RESULT IN A MARK OF ZERO (0) FOR THE LAB WORK MARK.