

**CSC111 Computer Science II**

**Lab 4 – Pass by Value & Pass by Reference**

**Introduction**

The purpose of this lab is to understand the difference of Pass by Value and Pass by Reference using `swap()` function. Once you finish Lab4, you will have at least four files under Lab 4 directory, lab4a.c, lab4b.c, lab4c.c, and lab4d.c.

**Getting started**

Using your favorite editor, set-up the standard format of C. Put the proper comments in each code.

```c
/* CSC111 Computer Science II
   Lab 4 Pass by Value & Pass by Reference
   Programmer: Your Name
   Professor: Dr. Lee
   File Created: February 19, 2019
   File Updated: February 19, 2019 */

Pass by Value: lab4a

#include <stdio.h>

void swap (int a, int b);

int main()
{
    int i = 1;
    int j = 2;
    printf("i = %d, j = %d\n", i, j);
    swap(i, j);
    printf("i = %d, j = %d\n", i, j);
    return 0;
}

void swap (int a, int b)
{
    printf("a = %d, b = %d\n", a, b);
    int temp = a;
    a = b;
    b = temp;
    printf("a = %d, b = %d\n", a, b);
}
```

In this example, while the values in `a` and `b` are swapped, the values of `i` and `j` are not. This is because the values of `i` and `j` are being passed by value – that means that there values are copied into `a` and `b`, but there is no other connection between `i` and `a`, or `b` and `j`. This is exactly the way you are used to seeing primitive values being passed in Java.
Pass by Reference: lab4b

In C, you can also choose to pass parameters by reference. You use this technique when you want the values of the variables being passed into a function to be changed. In order to do that, instead of giving the parameter the value, we want to make it point to the variable that we want changed.

This requires two new things:

1) We need to tell the compiler that the parameter is not going to be the primitive type, but is going to be a pointer to the primitive type. Therefore, the parameters include an asterisk before the variable name. Since those variables are now pointers, we need to also use the * to chase the pointer to its value when we use that variables. In this case, variable b will hold a pointer and *b will be the value at which that pointer is pointing.

2) Instead of passing the value of the variable to the function, we need to pass the address where that value is stored. We do this with the & operator.

```c
void swap (int *a, int *b);
int main()
{
    int i = 1;
    int j = 2;
    printf("before: i = %d, j = %d\n", i, j);
    swap(&i, &j);
    printf("after:  i = %d, j = %d\n", i, j);
    return 0;
}

void swap (int *a, int *b)
{
    printf("before: *a = %d, *b = %d\n", *a, *b);
    int temp = *a;
    *a = *b;
    *b = temp;
    printf("after:  *a = %d, *b = %d\n", *a, *b);
}
```

**Answer first and code later.**

lab4c.c
Pass i by value, pass j by reference:
i=   , j=
lab4d.c
Pass i by reference, pass j by value:
i=   , j=