CSC 215
Procedural Programming with C
Lab #5
Reading Arguments and Pointers

Tutorial Section

- Reading Arguments.
  - The main functions need to be written like this:
    - `main(int argc, char *argv[])`
  - int argc: Contain the number of arguments in argv.
  - char *argv[]: Is an array of strings (array of arrays of char).
    - Example: **`$./lab5 My Name`**
    - `argc will be 3.`
    - `argv[0] = "./lab5\0"`
    - `argv[1] = "My\0"`
    - `argv[2] = "Name\0"`
- Pointer's tutorial:
  - Declare an integer A and initialize it 10.
  - Declare an integer pointer ptr and let it point to A.
  - Print A, *ptr, &A, ptr, &ptr, &*ptr, and *&ptr.
  - Change A's value to 20.
  - Print A, *ptr.
  - Change *ptr's value to 30.
  - Print A, *ptr.
  - Declare an integer B and initialize it 5. Make pointer ptr point to it.
  - Print A, B, *ptr, &A, &B, ptr, &ptr, &*ptr, and *&ptr.
  - Declare an integer pointer ptr2 and let it point to A.
  - Create the method ***swap*** that takes two pointers and swaps the values (Not addresses).

```
void swap(int *num1, int *num2)
{
    int temp = *num1;
    *num1 = *num2;
    *num2 = temp;
}
```

  - Call swap and send ptr and ptr1 as parameters.
  - Print A and B.
  - Call swap and send &A and &B as parameters.
  - Print A and B.

```
----------------
A        = 10
*ptr     = 10
&A       = 2686728
ptr      = 2686728
&ptr     = 2686724
*&ptr    = 2686728
&*ptr    = 2686728

----------------
A        = 20
*ptr     = 20

----------------
A        = 30
*ptr     = 30

----------------
A        = 30
B        = 5
*ptr     = 5
&A       = 2686728
&B       = 2686720
ptr      = 2686720
&ptr     = 2686724
*&ptr    = 2686720
&*ptr    = 2686720

----------------
A        = 5
B        = 30

----------------
A        = 30
B        = 5
```

Lab Section

- Create the following functions:
    - A function called ***equals***. It will take two arrays of char (strings). Then returns 1 if they equal each other. 0 otherwise. **Use pointers only**. **Don't use string.h and don't declare any integers.**
        - `int equals(char Arr1[], char Arr2[])`
    - A function called ***convertNumber***. It will take an array of char (string) and a pointer to an integer. The string will contain the name of a number. The function should handle the following strings:
        - "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine".

      If the received string equals one of these strings, store the equivalent of that number in integer pointer result. Then return 1. Return 0 for any string that doesn't match the strings mentioned earlier.
        - `int convertNumber(char Arr[],int *result)`
        - **E.g.** If the string was "One", *result = 1.
    - A function called ***convertOperator***. It will take an array of char (string) and a pointer to a char. The string will contain the name of a mathematical operator. The function should handle the following strings:
        - "Plus", "Minus", "Multiply", and "Divide".

      If the received string equals one of these strings, store the equivalent of that operator in char pointer result. Then return 1. Return 0 for any string that doesn't match the strings mentioned earlier.
        - `int convertOperator(char Arr[],char *result)`
        - **E.g.** If the string was "Plus", *result = '+'.
    - A function called ***calculate***. It will take two integers, a char, and a pointer to float. The char will represent a mathematical operator. It must be '+', '-', '*', or '/'. The function should calculate the result of "num1 operator num2", then store the result in the float pointer result. Then return 1. If the operator was other than the previously mentioned chars, or if the user tries to divide by zero, then return 0.
        - `int calculate(int num1, int num2, char operator, float *result)`
- The ***main*** function. That should do the following:
    - Read the arguments and make sure that the user typed 3 arguments other than the program name. Print an error if that was not the case.
    - The user must use the following format:
        - $./ProgramName Number1 Operator Number2
    - Convert the numbers into integers and the operator into a char. If any of them was incorrect, print an error then stop the program.
    - Calculate the mathematical argument and show the result along with the numbers and the operator. Show an error when dividing by zero.

- Show your program to the instructor. Then upload it to LMS under Lab5 Homework.

- Example runs:

```
$./lab5 Five Plus Nine
5 + 9 = 14.00
$./lab5 Eight Multiply Six
8 * 6 = 48.00
$./lab5 Nine Divide Two
9 / 2 = 4.50
$./lab5 Three Minus Seven
3 - 7 = -4.00
$./lab5 Four Minus
ERROR: The program must read 3 arguments in the following order
"Number Operator Number".
$./lab5 Ali Plus Two
ERROR: The first number is incorrect.
$./lab5 Three Minus No
ERROR: The second number is incorrect.
$./lab5 Seven log Two
ERROR: The operator is incorrect.
$./lab5 Nine Divide Zero
ERROR: Can't divide by zero.
```

SUBMIT POLICY: -

- Use the follow naming convention: Lab05_ID_FirstName_LastName.c
  - **Example:** Lab05_123456789_Marwan_Almaymoni.c
- Use a comment to write your name and ID at the beginning of the code.
- The Deadline is: 23/03/2015 right before the Lab starts.
- Late submissions will not be accepted.
- Email submissions will not be accepted.
- **-1 Point** for not following the naming convention.
- **-1 Point** for not writing your name and ID in the code inside a comment.
- **-8 Points** if the submitted program didn't work due to syntax errors.
- **-10 Points** for cheating and helping others cheat.