**Purpose:** In this assignment we will be learning how to write macros.

To begin, download the provided assignment 4 template file. The file contains empty declarations for four macros along with a _start function to test them.

Do not alter the _start function in any way. Fill out each macro to pass the tests. All macro invocations use only variable names or memory segments. Feel free to use any registers you want in your macros.

**Instructions**

To complete this assignment, you must write the code in each macro to make them work.

**getStringLength** (arguments 2): Get string length will take a null terminated string and determine how many non-null characters appear before the null. Return the length in the second argument..

The string

"This is an example", LINEFEED, NULL

is length 19.

**getDays** (arguments 5): This macro will take a number of seconds and break it down into an equivalent number of days, hours, minutes, and seconds. For example, 3671 seconds would equal:

0 Days, 1 Hours, 1 Minute, 11 Seconds

Use division and remainders to calculate the values. The provided constants are to calculate from seconds, then minutes, hours, and finally days.

3671 / 60 = 61 minutes with remainder 11 seconds
61 / 60 = 1 hour with remainder 1 minute
1 / 24 = 0 days with remainder 1 hour

**stringToInt32** (arguments 2): This macro will convert a string containing a whole number value to its equivalent as a signed dword integer.

Assume the number will come in the following format:
    The first character may contain '+' or '-'
    At least 1 character '0' through '9'. (max 10)
    A NULL character
Converting a character to its numeric value:
    sub byte[character], '0'
Use the following algorithm to calculate the value:
    Start with a sum of 0.
    For each digit character:
        multiply by 10
        Add the numeric value of the digit
    Multiply by the sign
        No Sign or '+': Multiply by 1
        '-': Multiply by -1

You do not need to do any error checking on the inputs, assume that they will be entered correctly.

**Example**

```
    Input: "-910", NULL

    Check first character: '-' found.  Value is negative.
    Sum = 0
    Check next character: '9' digit found.
        Convert '9' to 9.
        sum = sum * 10 + 9
        sum = 9
    Check next character: '1' digit found.
        Convert '1' to 1.
        sum = sum * 10 + 1
        sum = 91
    Check next character: '0' digit found.
        Convert '0' to 0.
        sum = sum * 10 + 0
        sum = 910
    Check next character: NULL found.
        Apply sign to sum: -910
```

**int32ToString** (arguments 2): This macro will convert a 32 bit signed integer to a string.

The algorithm is very similar to the method to convert between decimal and binary numbers.

Determine if the value is negative. If it is, multiply the value by -1.

Dividing the value by 10 will generate the 1's digit and slide over the rest of the value.

Use the stack to push and pop the generated digits as you will want to write them into the string from left to right.

Include a negative sign if the value was negative.

Place a NULL character after the final digit.

**Example**

Input: -416

Remember that the value is negative and multiply by -1.

416 / 10 = 41 r 6, push 6 to stack
41 / 10 = 4 r 1, push 1 to stack
4 / 10 = 0 r 4, push 4 to stack

Since value was negative, put '-' into string.

"-"

Pop each digit and add to string.

"-4"
"-41"
"-416"

Add a null at the end.

"-416", NULL

## Submission

Once you are satisfied with the program, upload the assembly source code (.asm) file to the assignment #2 page on the class website.

## Expected Outcome:

```
getStringLength: PASS
getDays: PASS
String to Integer 1 (-41632): PASS
String to Integer 2 (9055): PASS
String to Integer 3 (+3): PASS
Integer to String (-88,216): -88216
Integer to String (372,441): 372441
```