Purpose: This program will involve working with macros to perform a variety of tasks related to text manipulation.

Download the provided template file, it contains some provided data as well as code to output to the console.

You will not need to alter the _start section of the program. All of your code will go in the macros at the top of the program.

Program Specifications
You will need to write the following macros:

1. A macro that will take a string as an argument in the form of a memory address to a byte array. The macro must count the number of characters (including null) in the string and store it in the rax register. The macro should stop when it encounters the null character.

```
exampleString db “Hello, World!”, NULL
```

The macro should return 14 in rax if given exampleString.

2. A macro that takes a single memory address as an argument. This macro will need to convert any lowercase letters in the string to uppercase. The macro should stop when it encounters the null character.

```
OriginalString db “Testing, testing. 1, 2, 3.”, NULL
```

After the macro call, OriginalString should be changed to

“TESTING, TESTING. 1, 2, 3.”, null.

3. A macro that takes two arguments, a memory address to a dword variable and a memory address to a string. The string will contain a signed whole number decimal value as text. The macro will calculate the value of the decimal string and store it in the provided variable as a signed integer.

Hint: The text will be formatted as follows:
0 or more spaces
A ‘+’ or ‘-’ may appear (assume positive if no sign)
0 or more spaces
1 or more numerals (‘0’ to ‘9’)
0 or more spaces
A NULL character

Algorithm:
```
value = 0
for each digit starting from the left
    value = value x 10
    value = value + digit
if (sign == ‘-’) value = value * -1
```

Example:
“-914”

Sign: -

value = 0
value = value x 10 = 0
value = value + 9 = 9
value = value x 10 = 90
value = value + 1 = 91
value = value x 10 = 910
value = value + 4 = 914

value = value x -1 = -914

4. A macro that takes two arguments, a memory address to a dword variable and a memory address to an array of 11 bytes. The dword will contain a signed integer value. The macro will need to convert the dword value to a hexadecimal string representation stored in the byte array.

The hexadecimal string must include “0x” before the hexadecimal digits and must end with a null character.

All values generated must be exactly 8 hexadecimal digits long (including any leading 0’s). This may require adding extra digits to the string.

Use capital letters for the hexadecimal values 10-15.

Algorithm:

```
repeat until value = 0
    divide value by 16
    if remainder is between 0 and 9, convert to ‘0’ - ‘9’
    else convert to ‘A’ - ‘F’
    set byte in array to converted character
    move to previous byte in array
```

The remainders will generate values from right to left. Start at the character just before the NULL and work to the left.

Use unsigned division.

Hint:

```
    ‘A’ = 65
    ‘0’ = 48
    0 + 48 = ‘0’
    1 + 48 = ‘1’
    10 + 55 = ‘A’
    15 + 55 = ‘F’
```

Example:

```
    Integer 252 would convert to “0x000000FC”, NULL
    Integer  -3 would convert to “0xFFFFFFFD”, NULL
```

Expected Output:

```
Macro 1: Pass
Macro 2: DID YOU READ CHAPTERS 8, 9, AND 11 YET?
Macro 3-1: Pass
Macro 3-2: Pass
Macro 3-3: Pass
Macro 4-1: 0x000000FF
Macro 4-2: 0x001E582A
Macro 4-3: 0xFFFFFFF9
```

Submission

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