Robin Ward
COMP 7720/7726/4970
Summer 2018
5/31/2018
Auburn University

1.
    a. The purpose of the uplow_lowup is to change the case of the array. Specifically change the case to the opposite.
    b. The call convention used by uplow_lowup is __fastcall. Lines 37-41 indicate this in the assembly code.
    c. I counted 8 in the assembly code for that uplow_lowup function. They are at lines 50, 56, 61, 72, 79, 86, 91, 102.
    d. There are 2 local variables: i, count. Lines 43 and 45 in the assembly support this.
    e. The contents of ecx will be ecx plus the addition of the bytes of memory at memory address letters.
    f. The purpose of the lines 48-50 are to initiate the while loop, “while (i < size)”.
    g. Eax. this will get set at line 116.
    h. The first argument pushed to the stack is size. This is defined at line 28. It is used at the following lines in the assembly code: 40, 49
    i. I do not believe that the assembly is the best way to implement this C code. The reason is mainly because of simplicity. The C code is much easier to write and understand. It is also a lot easier to pass off to another developer for review because more people know C than assembly. Another reason is because there are compilers that can turn C code into assembly in a matter of seconds. So if the user needed to access the assembly, they could write it in C then decompile it.
2. The main reason why a lot of this code will change is due to the call change. The main difference is that fastcall puts two parameters into the register instead of one. Integer values returned to the register are EAX. we are also changing from the int to unsigned short, which is a short unsigned integer type. This data type cannot be negative. Listed below are the assembly lines of code that will change. These are all changed due to the data type size by changing the data type to short.

    - **G**
    - **H**
    - **I**
    - **K**
    - **M**
    - **O**
    - **P**