CSCI 2330 – x86-64 Assembly Exercises

1. For each of the following x86-64 instructions, rewrite as a C-style command using assignment (=), pointer dereferencing (*), and regular arithmetic operators (+, –, etc). You can use register names as subexpressions – e.g., "**`movq %rax, %rcx`**" could be rewritten as "**`rcx = rax`**". Assume no data size scaling for C pointer arithmetic.

(a) **`addq %rax, %rcx`**

(b) **`movq %rax, (%rcx)`**

(c) **`subq (%rax), %rcx`**

(d) **`leaq (%rax), %rcx`**

(e) **`leaq 9(%rax, %rdx), %rbx`**

(f) **`addq 9(%rax, %rdx), %rbx`**

2. Assuming **`func1`** and **`func2`** are non-void functions that each take two arguments, rewrite the following x86-64 instructions as a series of C-style function calls (such as "**`int x = someFunc(20, 30);`**"). Assume that the size of a **long** is 8 bytes. You can define and use any variables you wish.

```
movq    $5, %rsi
movq    $8, %rdi
callq   func1
movq    %rax, %rsi
callq   func2
movq    %rax, %rdi
callq   func1
```

3. Consider the x86-64 instructions below for a function named **`check`**. Rewrite this code as a C function, which you can assume takes two **int** arguments and returns an **int**. You can use any variable names desired in your C function.

```
check:
    subl    $4, %edi
    cmpl    %edi, %esi
    setl    %al
    movzbl  %al, %eax
    ret
```