CSCI 2330 – (More) x86-64 Assembly Exercises

1. Translate the x86-64 instructions below into a C function, which you can assume is non-void and takes one argument. Remember to specify the appropriate types of the argument and return value. **Do not use any goto statements in your function.**

```
compute:
     movq      $0, %rax
     movq      $1, %rbx
     jmp       .L2
.L1: addq      $1, %rax
     salq      $1, %rbx        # left shift
.L2: cmpq      %rdi, %rbx
     jl        .L1             # jump if less
     ret
```

2. Translate the x86-64 instructions below into a C function, which again is a non-void function with one argument. As before, do not use any goto statements. This function uses data in memory; the specific addresses used are not significant (and can't be determined here anyways), but what each location is used for does matter. Keep track of which locations are used. **Hint:** the argument is a pointer type!

```
sum10:
     pushq     %rbp                # disregard for now
     movq      %rsp, %rbp          # disregard
     movq      %rdi, -0x18(%rbp)   # copy %rdi to some address
     movl      $0x0,-0x4(%rbp)
     movl      $0x0,-0x8(%rbp)
     jmp       .L2
.L1: movq      -0x18(%rbp),%rax
     movl      (%rax),%eax         # pay close attention here!
     addl      %eax,-0x4(%rbp)
     addq      $0x4,-0x18(%rbp)
     addl      $0x1,-0x8(%rbp)
.L2: cmpl      $0x9,-0x8(%rbp)
     jle       .L1                 # jump if less or equal to
     movl      -0x4(%rbp),%eax
     popq      %rbp                # disregard
     ret
```