Practice Problem 3.25:

For C code having the general form

```
long loop_while2(long a, long b)
{
  long result = ____________;
  while (_______________) {
    result = _____________;
    b = ____________;
  }
  return result;
}
```

GCC, run with command-line option `-O1`, produces the following code:

a in %rdi, b in %rsi

```
   1.  loop_while2:
   2.    testq     %rsi, %rsi
   3.    jle       .L8
   4.    movq      %rsi, %rax
   5.  .L7:
   6.    imulq     %rdi, %rax
   7.    subq      %rdi, %rsi
   8.    testq     %rsi, %rsi
   9.    jg        .L7
   10.   rep; ret
   11. .L8:
   12.   movq      %rsi, %rax
   13.   ret
```

We can see that the compiler used a guarded-do translation, using the `jle` instruction on line 3 to skip over the loop code when the initial test fails. Fill in the missing parts of the C code. Note that the control structure in assembly code does not exactly match what would be obtained by a direct translation of the C code according to our translation rules. In particular, it has two different ret instructions (lines 10 and 13). However, you can fill out the missing portions of the C code in a way that it will have equivalent behavior to the assembly code.

Practice Problem 3.35:

For a C function having the general structure

```
long rfun(unsigned long x) {
  if (_________________)
    return ____________;
  unsigned long nx = _____________;
  long rv = rfun(nx);
  return ________________;
}
```

GCC generates the following assembly code:

```
long rfun(unsigned long x)
x in %rdi

 rfun:
   pushq     %rbx
   movq      %rdi, %rbx
   movl      $0, %eax
   testq     %rdi, %rdi
   je .L2
   shrq      $2, %rdi
   call      rfun
   addq      %rbx, %rax
 .L2:
   popq      %rbx
   ret
```

A. What value does `rfun` store in the callee-saved register `%rbx`?

B. Fill in the missing expressions in the C code shown above.