Section 4
Assembly and GDB

1. Assembly to C: What does the following code do?

```
iii)        movl    (%rdi), %eax
            leal    (%eax,%eax,2), %eax
            addl    %eax, %eax
            andl    %esi, %eax
            subl    %esi, %eax
            ret
```

(14au midterm)

2. C to Assembly:

Given the following C function:

```
long happy(long *x, long y, long z) {
  if (y > z)
    return z + y;
  else
    return *x;
}
```

Write **x86-64** bit assembly code for this function here. Comments are not required but could help for partial credit. We are not judging you on the efficiency of your code, just the correctness. It is fine to leave off the size suffixes if you prefer to (e.g. b, w, l, q).

(15au midterm)

Tutorial Script For Phase 1

```
gdb bomb
break explode_bomb
break phase_1
break finish_lab      (this function doesn't exist)
run
[input a string]
disas                 (shows disassembly of phase 1, also your current place
                      in the program)
help info             (illustrate help command)
info registers        (show the contents of the registers)
q
step                  (bomb will explode now unless you magically guessed the
                      right string)
kill                  (will hit breakpoint on explode_bomb, don't want it to
                      explode, kill it instead!)
run
[input a string]
stepi
stepi
disas                 (show that we are at the function call to compare
                      strings--layout asm is ok too)
x /10wx $rdi          x /NUM SIZE FORMAT y (shows contents of memory at
                      address y)
x /s $rdi             (hey looks like when we interpret the contents as
                      characters...)
x /s $rsi             (let's look at what is in the other register... hey!)
kill
run
[your string for phase 1!]
```