ECEn 224: Assembly Homework

For all questions, you should compile the C code you come up with for the function and try to match the assembly code exactly. Use the Digital Lab computers to compile your code with the following gcc flags: `gcc -Og -S <name_of_source_file>`. For each problem, create a C file with the name, `pX.c`, where X is the number of the question. So for question 1, your answer will be in `p1.c`. You will need to write a main function that calls the mystery function as well as filling out the mystery function. To turn in the lab, zip up all of the source files together and upload it to Learning Suite.

1. (1 point) Fill in the function using the following assembly code.

```
int mystery(/* Add arguments here */) {
    /* Put something here */
}

mystery:
    addl        %edi, %esi
    leal        (%rsi,%rsi,2), %edx
    leal        0(,%rdx,8), %eax
    ret
```

2. (1 point) Fill in the function using the following assembly code.

```
int mystery(int x) {
    /* Put something here */
}

mystery:
    testl   %edi, %edi
    je      .L3
    movl    %edi, %eax
    sall    $4, %eax
    ret
.L3:
    movl    $5, %eax
    ret
```

3. (1 point) Fill in the function using the following assembly code.

```
long mystery_loop(long start, long max) {
    /* Put something here */
}
```

```
mystery_loop:
    movq    %rdi, %rax
    movl    $0, %edx
.L2:
    addq    %rax, %rax
    addq    $1, %rdx
    cmpq    %rsi, %rdx
    jl      .L2
    rep ret
```

Note: On the last line, ignore the `rep` and treat it as just a `ret`. If you are interested, here is a Stack Overflow question about it.

4. (3 points) Fill in the function using the following assembly code.

```
typedef struct {
    long x;
    long y;
} point_t;

void mystery_struct(point_t *point) {
    /* Put something here */
}
```

```
mystery_struct:
    movq    (%rdi), %rax
    testq   %rax, %rax
    jle     .L2
    movq    8(%rdi), %rdx
    movq    %rdx, (%rdi)
    movq    %rax, 8(%rdi)
    ret
.L2:
    cmpq    $0, 8(%rdi)
    jle     .L4
    addq    $9, %rax
    movq    %rax, 8(%rdi)
    ret
.L4:
    movq    $0, (%rdi)
    movq    $0, 8(%rdi)
    ret
```

5. (6 points) Fill in the switch statement using the following assembly code.

```
typedef enum { MODE_A, MODE_B, MODE_C, MODE_D, MODE_E } mode_t;

long mystery_switch(long *p1, long *p2, mode_t action) {
    long result = 0;
    switch (action) {
    case MODE_A:
        /* Put something here */

    case MODE_B:
        /* Put something here */

    case MODE_C:
        /* Put something here */

    case MODE_D:
        /* Put something here */

    case MODE_E:
        /* Put something here */

    default:
        /* Put something here */
    }
    return result;
}
```

```
.L8:                            # MODE_E
  movl    $27, %eax
  ret
.L3:                            # MODE_A
  movq    (%rsi), %rax
  movq    (%rdi), %rdx
  movq    %rdx, (%rsi)
  ret
.L5:                            # MODE_B
  movq    (%rdi), %rax
  addq    (%rsi), %rax
  movq    %rax, (%rdi)
  ret
.L6:                            # MODE_C
  movq    $59, (%rdi)
  movq    (%rsi), %rax
  ret
.L7:                            # MODE_D
  movq    (%rsi), %rax
  movq    %rax, (%rdi)
  movl    $27, %eax
  ret
.L9                             # default
  movl    $12, %eax
  ret
```