Part 6: Laboratory Questions

1) In symbol_table.c, why does the char *types[] array have a bunch of empty strings?

The #defines are based on the size of the elementary data type size so the strings are offset by 1, 2, 4, 8.

2) What error is generated when using a floating-point constant for declaring an array size? Example: `int b[1.1];`

error: size of array 'b' has non-integer type

3) In the function new_symbol, why are the sizeof the type specifiers (2, 4, 8) installed into the symbol table?

The values are used in the print_quad function to print the multiple (*) instructions to determine the correct index into the array.

4) Can you use a storage class type to declare a variable? Example: `extern extern1;`

Yes!

5) Can you use the sizeof() operator on a storage class type? Example: `sizeof( extern)`

No!

Exercise 8.2.1: Generate PIC 16F84 code for the following three-address statements assuming all variables are stored in memory locations.

a) x = 1

```
movlw 1
movwf x
```

b) x = a

```
movf    a, w
movwf  x
```

c) x = a + 1

```
movlw 1
addwf a, w
movwf  x
```

c) x = a + b

```
movf b, w
addwf a, w
movwf  x
```

Exercise 8.2.2: Generate PIC 16F84 code for the following three-address statements assuming a and b are arrays whose elements are 1-byte values.

a) x = a[i]

```
movf i, w          ; w = i
addlw a            ; w =  a + w
movwf FSR
movf INDR, w
movwf x            ; x = w
```

y = b[j]

```
movf j, w          ; w = j
addlw b            ; w =  b + w
movwf FSR
movf INDR, w
movwf y            ; y = w
```

a[i] = y

```
movf i, w         ; w = i
addlw a           ; w = a + w
movwf FSR
movf y, w         ; w = y
movwf IND
```

b[j] = x

```
movf j, w         ; w = j
addlw b           ; w = b + w
movwf FSR
movf x, w         ; w = x
movwf IND
```