ME333 – Homework 1 – Yael Ben Shalom

Exercise 18:

Invoking the gcc compiler with a command like "gcc myprog.c -o myprog" initiates 4 steps:

1. **Preprocessing** - The preprocessor takes the program.c source code and produces an equivalent .c source code.
2. **Compiling** - The compiler turns the preprocessed code into assembly code for the specific processor.
3. **Assembling** - The assembler converts the assembly instructions into processor-dependent machine-level binary object code.
4. **Linking** - The linker takes one or more object code files and produces a single executable file.

Exercise 19:

The function main's return type in written before the function name. That means that the function should end by returning a value of that type. For example, the return type for "int main(void)" will be int, for "float main(void)" will be float, and for "void main(void)" will be void (nothing).

Exercise 21:

```
unsigned char i = 60, j = 80, k = 200;
unsigned char sum;  // 0 - 255
```

(a) `sum = i+j = 60 + 80 = 140;  // (140 < 255)`

(b) `sum = i+k = 60 + 200 = 4;  // (260 > 255)`

(c) `sum = j+k = 80 + 200 = 24;  // (280 > 255)`

Exercise 22:

```
int a = 2, b = 3, c;
float d = 1.0, e = 3.5, f;
```

(a) `f = a/b = 2/3 = 0;`

(b) `f = ((float) a)/b = 2.0/3 = 0.66;`

(c) `f = (float) (a/b) = (float) (2/3) = 0;`

(d) `c = e/d = 3.5/1.0 = 3.5;`

(e) `c = (int) (e/d) = (int) (3.5) = 3;`

(f) `f = ((int) e)/d = 3/1.0 = 3.0;`

Exercise 27:

The systematic strategy for debugging in to debug every function separately, and print the variable values to the screen after every step. Especially pay attention to data sizes and math operation, and check the values for overflow and information loss.

Exercise 28:

The comment-less invest.c is attached to this PDF.

Exercise 30:

```
int x[4] = {4, 3, 2, 1};
```

(a) `x[1] = 3;`

(b) `*x = 4;`

(c) `*(x+2) = 2;`

(d) `(*x)+2 = 6;`

(e) `*x[3] = 1;`

(f) `x[4] = unknown;`

(g) `*(&(x[1]) + 1) = 2;`

Exercise 31:

```
int i,k=6;
i = 3*(5>1) + (k=2) + (k==6);
```

I would think that `(k=2)` is an assignment and not a value, and therefore i = error, but I run the line online and the result was:

`i = 3*1 + 1 + 0 = 4;  // (5>1) = True = 1; (k=2) = assignment = 1; (k==6) = False = 0 (k=2 now).`

Exercise 32:

```
unsigned char a=0x0D, b=0x03, c;            // a = 0b00001101, b = 0b00000011
```

(a) `c = ~a;  // c = 0b11110010 = 0xF2`

(b) `c = a & b;  // c = 0b00000001 = 0x1`

(c) `c = a | b;  // c = 0b00001111 = 0xF`

(d) `c = a ^ b;  // c = 0b00001110 = 0xE`

(e) `c = a >> 3;  // c = 0b00000001 = 0x1`

(f) `c = a << 3;  // c = 0b01101000 = 0x68`

(g) `c &= b;  // c = c & b = c & 0x03`

Exercise 34:

The ASCII code is attached to this PDF.

Exercise 35:

The bubble sort code is attached to this PDF.