Plan 2

September 16 – September 23

- Read Chapter 3 in the textbook. Skip Section 3.6.6.

- **Exercise 2**

  Implement a C program for printing the 32-bit representation (`float`) of -3.1415. Check that its output is correct according the IEEE-754 single precision floating-point standard.

  Hint: Logical operations cannot be applied on `float` types. A `float f` may be represented in an `int i` by executing `i = *(int *) &f`.

- **Exercise 3**. Solve the exercise on the next two pages.

Exercise 3

The C program below reads two 16-bit integers and prints their product.

```
#include <stdio.h>

int mult(short a, short b) {
    int ia = a, ib = b;
    return ia * ib;
}

int main() {
    short a, b;
    printf("Enter two integers: ");
    scanf("%hd %hd", &a, &b);
    printf("Product = %d\n", mult(a, b));
}
```

1. Compile and run the program under Unix/Linux.

2. Assuming the input integers are non-negative, implement the `mult` function using the traditional pencil and paper method for binary numbers. Hint: This will require some bit manipulation operations. You are going to use & (the bitwise AND operator), << (the "shift left" operator), and >> (the "shift right" operator).

3. Extend the solution of question 2 so that the `mult` function can handle negative integers as input.

4. Implement `mult` using Booth's algorithm.

5. Compare the runtime efficiency of the four implementations of `mult`.