1. (4 pts) (K&R Ch 1-4)
What is the output of the following C code?

```c
main()
{
    int i = 6;
    int j = -35;
    printf("%d %d\n", i++, ++j);
    i = i << 3;
    j = j >> 4;
    printf("%d %d\n", i, j);
}
```

```
6 -34
56 -3
```

2. (2 pts) (B&O Ch. 1,7)
a) What style of linking produces binaries that are self-contained and contain no references to code in the file system?

```
static
```

b) Which step in the compilation process will take C programs and produce expanded C programs for the compiler?

```
pre-processor
```

3. (4 pts) (B&O Ch. 7, Problem 7.1)
Consider the following program:

```c
int init=5;
int x;
main() {
    int y=0;
    y = x+init;
    return y;
}
```

a. What section of the binary would contain variable x?

```
BSS
```

b. What section of the binary would contain the code for main?

```
Text
```

4. (4 pts) (B&O Ch. 2.1, Problem 2.4)
a) $0x637a + 0x3a =$

```
a) 637a
   3a
   6384
```

b) $0x63a0 - 0x45 =$

```
b) 63a0
   45
   6358
```
5. (12 pts) (B&O Ch. 2.1, Problems 2.1, 2.3)
   a) Convert 153 from decimal to binary
      \[ 10011001 \]
   b) Convert AE from hexadecimal to binary
      \[ 10101110 \]
   c) Convert 186 from decimal to hexadecimal
      \[ 1011100 = BA \]
   d) Convert 10101110 from binary to hexadecimal
      \[ \text{AE} \]
   e) Convert 01011011 from binary to decimal
      \[ 91 \]
   f) Convert DA from hexadecimal to decimal
      \[ 218 \]

6. (2 pts) (B&O Ch. 2.1, Problem 2.5)
   Consider this program:
   ```c
   #include <stdio.h>
   int main() {
     int i=0x40302010;
     unsigned char *cp;
     cp = (unsigned char *) &i;
     printf("%x\n", *cp);
   }
   ```
   a) What is its output on a little endian machine?
      \[ 0x10 \]
   b) What is its output on a big endian machine?
      \[ 0x40 \]

7. (4 pts) (B&O Ch. 2.1, Problem 2.12)
   Assuming x86-64, write a single C expression that takes a value \( x \) and returns \( x \) with its least significant two bytes set to 0. Use only the variable \( x \) and bit-wise operators. (i.e. Do not use `=`)
   \[ x \& (\sim 0xffff) \]
   \[ (x \& 0xffff) \sim x \]
8. (10 pts) (B&O Chapter 2.1, Problem 2.8, 2.14)
Fill in the result of the following expressions assuming the following declaration.

```
unsigned char a=0xB5;
unsigned char b=0x36;
unsigned char c=0x00;
```

Give all answers in hexadecimal notation. Note that logical operations return 0x1 or 0x0.

a) \(( a \& b )\) 0x34

b) \(( a ^ b )\) 0x83

c) \(( a \| b )\) 0x1

d) \(\sim c\) 0xff

e) \(!c\) 0x1

9. (16 pts) (B&O Chapter 2.2, Problem 2.17, 2.19, 2.22)

a) Represent the number \(-5\) in a 4-bit two’s complement format

b) Represent the number \(5\) in a 4-bit two’s complement format

c) Consider the 5-bit two’s complement number \(10110\), what is its decimal value?

d) Consider the 5-bit unsigned number \(10110\), what is its decimal value?

e) Give the hex representation of the largest positive 32-bit two’s complement number.

f) Give the hex representation of the most negative 32-bit two’s complement number.

g) Write the hexadecimal value of the 8-bit signed integer \(-13\)

h) Write the hexadecimal value of the 32-bit signed integer \(-13\)
10. (4 pts) (B&O Chapter 2.2, Problem 2.21)
For expressions that mix signed and unsigned numbers, C will cast the signed value to an unsigned one before evaluation. In C, list whether the following expressions are true or false.
   a) \((0U < -1)\)
      \[\text{True}\]
   b) (unsigned) \(-3 > -35\)
      \[\text{True}\]

11. (4 pts) (B&O Chapter 2.2, Problem 2.23)
For these 32-bit data objects:
   \[
   \begin{align*}
   \text{int } x &= 0x88888888; \\
   \text{unsigned int } ux &= 0x88888888;
   \end{align*}
   \]
   a) What is the hexadecimal value of \((x << 20) >> 20)\)?
      \[0xFFFF_F888\]
   b) What is the hexadecimal value of \((ux << 20) >> 20)\)?
      \[0x0000_888\]

12. (4 pts) (Chapter 2.2, Problem 2.26)
Type errors can cause problems in programs. One common bug relates to the mixing of unsigned data types like size_t with signed integer types. With this in mind, what is the output of the following program:
   ```c
   #include <string.h>
   /* size_t strlen(const char* str); */
   int strshorter(char *s, char *t) {
     return (strlen(s) - strlen(t)) < 0;
   }
   main() {
     if (strshorter("foo","bar"))
       printf("foo < bar\n");
     if (strshorter("bar","food"))
       printf("bar < food\n");
     if (strshorter("food","bar"))
       printf("food < bar\n");
   }
   ```
   \[\text{No output}\]
13. (6 pts) (B&O Chapter 2.3, Problem 2.29)
a) What is the decimal value of the sum of the following 6-bit two’s complement numbers? \( 100110 + 100101 \)

\[
\begin{array}{c}
\begin{array}{c}
00010111 \\
00100101 \\
\hline
101101
\end{array}
\end{array}
\]

b) What is the decimal value of the sum of the following 6-bit two’s complement numbers? \( 111101 + 011101 \)

\[
\begin{array}{c}
\begin{array}{c}
01110110 \\
00101010 \\
\hline
1011110
\end{array}
\end{array}
\]

c) What is the decimal value of the sum of the following 6-bit two’s complement numbers? \( 011001 + 011101 \)

\[
\begin{array}{c}
\begin{array}{c}
00100101 \\
00111110 \\
\hline
1010001
\end{array}
\end{array}
\]

14. (4 pts) (Chapter 2.3, Problem 2.40)
Suppose we are given the task of generating code to multiply integer variable \( x \) by various different constant factors \( K \). To be efficient we want to use only the operations \(+, -,, and <<\). For the following values of \( K \), write C expressions to perform the multiplication using at most three operations per expression.
a) \( K=63 \)

\[
(x << 6) - x
\]
b) \( K=48 \)

\[
(x << 5) + (x << 4)
\]

15. (4 pts) (Chapter 2.4, Problem 2.45)
a) Write the following fraction as a binary number using a binary point \( \frac{27}{32} \).

\[
0.11001
\]
b) Write the fractional value of the following binary number \( 11.1011 \)

\[
3\frac{11}{16} = \frac{59}{16}
\]

16. (4 pts) (Chapter 2.4, Problem 2.54)
Assume variable \( i \) of type int. For the following C expressions, state whether it will always be true or give a value such that it is not true.
a) \( i == (\text{int}) (\text{float}) i; \)

\[
\text{false for integers } > 2^{23}
\]
b) \( i == (\text{int}) (\text{double}) i; \)

\[
\text{True}
\]
17. (12 pts) (Chapter 2.4, Problem 2.47)
Consider an IEEE-based floating point format below with one sign bit, four exponent bits, and two fraction bits. The exponent has a Bias of 7. Recall, an exponent of all 0s denotes a denormalized number while an exponent of all 1s denotes infinite/NaN values.

+--------------------------------------------------+
| s | e3 e2 e1 e0 | f2 f1 f0 |
+--------------------------------------------------+

a) Give the bit-representation of the smallest, non-zero, positive number in this format.

\[
\begin{array}{c}
00000001;
\end{array}
\]

b) What is the value of this number given as a fraction?

\[
2^{-6} \times \frac{1}{8} = \frac{1}{512}
\]

c) Give the bit-representation of the largest, non-infinite, positive number in this format.

\[
01111111
\]

d) What is the value of this number?

\[
2^{7} \times \frac{1}{8} = \frac{15}{8} \times 2^7 = 15 \times 128 = 240
\]

e) In this format, calculate the value the following bit representation: 0 0000 101

\[
2^{-6} \times \frac{5}{8} = \frac{5}{512}
\]

f) In this format, calculate the value the following bit representation: 0 1010 111

\[
2^{3} \times \frac{7}{8} = 2^{3} \times \frac{15}{8} = 15
\]