Question 1:

17=16+1=2^4+2^0
Base 2 10001
Base 8 010 001=21 base 8
Base 16 0001 0001 = 11 Base 16

FF base 16 = 1111 1111 base 2
011 111 111 =377 base 8
$\sum_{n=0}^{7} 2^n = 255$ Decimal

Question 2:

125 base 10 is computed using the method described in class (dividing by 2 and if remainder is present the coefficient is 1. Note that the C++ code can also be used to check your division. This will lead to 0111 1101 base 2 and 7D base 16.

107 base 10 using the same process we get 0110 1011 base 2 and 6B base 16

Question 3:

34 base 5 is $3(5)+4(1)=19\, base\, 10$. This is the same as writing: 2^4+2^1+1 which is 10011 base 2 and 010 011 which is 23 base 8.

Question 4:

67 base 10 is 43 base 16 and is 0100 0011 base 2. Note Most Significant Bit (MSB) is the sign bit.
94 base 10 is 5E base 16 and is 0101 1110 base 2. (MSB is the sign bit).
94 one’s complement is 1010 0001 (inverted bits)
94 two’s complement is 1010 0010 (add one)
Now we have:
0100 0011
1010 0010

1110 0101
This is a negative number but what number?
0001 1010
0001 1011 (23 base 10)
So 1110 0101 is -23 using two’s complement (8 bit number)
This agrees with the decimal math.

Question 5:

D2 base 16 is a signed integer in sign-magnitude format.
1101 0010
-ve 101 0010 this is 64+16+2= 82 so this is -82.

Question 6:

```
(a) 26 = 0 000011010

   -37 = 1 000100101

  -123 = 1 001111011


(b) 26 = 0 000011010         ( note that positive numbers have the
                               same representation in sign-magnitude
                               and two's comp. )

  -37 = 1 111011011

 -123 = 1 110000101
```

Question 7:

```
(i)  010110       (+22)
   + 001001      +(+ 9)
     ------      ------
     011111      +31

(ii)   011001     (+25)
     + 010000   + (+16)
      -------     -----
       101001      (+41)

       Overflow has occurred

(iii)  010110     (+22)     =       010110
     - 011111   - (+31)        + two's comp(011111)
                 ------
                   -9

                 =  010110
                  + 100001
                   --------
                    110111  ===> -9 in twos comp
```

```
         1s comp =  001000
                  +      1
                   -------
                    001001
```

Question 8:
These calculations are the same as those discussed previously.
OR
Divide by 16
2748 Remainder #1 = 13/16, 171 Remainder #2=12/16, 10 Remainder 11/16, Last division yields 0 Remainder 10/16. Since 10 is A, 11 is B, 12 is C and 13 is D we can write:

43981 base 10 = ABCD Base 16

We can use the previously discussed methods OR
For 0.1972 we can multiply by 16 3.1552 so 3 base 16 with 0.1552, then 2 base 16 with 0.4832, then 7 base 16 with 0.7312, then B base 16 with 0.6992, then B with 0.1872, then 2 base 16 with 0.9952. Since we only need 12 bits 0.327 base 16 will be sufficient 0.0011 0010 0111 (this will have round off errors and be equivalent to 0.19702 base 10).

Question 9:

The first number is 1010 1011 1100 1101 so we must shift:
1.010 1011 1100 1101 (note that the mantissa is 23 bits and we are only using 15 so padding with zeros). This will lead to a mantissa of:
1.010 1011 1100 1101 0000 0000. The leading 1 is omitted so we can write 0.010 1011 1100 0000 0000 for the mantissa. We shifted the number 15 times to the right, this means the exponent is 15. Excess 127 will make this number 142 base 10 or 1000 1110 base 2. The number is positive so the sign bit is 0.
0 1000 1110 010 1011 1100 1101 0000 0000 is the number in binary format (leading 1 not shown but is there when you use the number for calculations).

For hex we use groups of 4 binary numbers so
0100 0111 0010 1011 1100 1101 0000 0000 which is 472BCD00 hex

The fraction is 0.0011 0010 0111 (using only the 12 bits from previous problem but you should actually use up to 23 bits so you should use red bits shown here 0.0011 0010 0111 1011 1011 0010) .
This will require three shifts to obtain: 1.1 0010 0111. In this case the mantissa will still require 23 bits and we must pad with zeros (this is a loss in accuracy and why we should use more than 12 bits).
1.10010 0111 0000 0000 0000 00
This will lead to a mantissa of
1001 0011 1000 0000 0000 000.

We had to shift three times to the right, this means that the exponent is -3, which in excess 127 notation is 124. In binary this is 0111 1100. The number is positive so the sign bit is zero. The number in binary IEEE 754 32-bit floating point format is:

0 0111 1100 1001 0011 1000 0000 0000 000

In hex it is typical to group the bits in fours.

0011 1110 0100 1001 1100 0000 0000 0000

Which is 3E49C000 in hex.

On your own try with the extra bits (shown in red above) and see what the difference is in terms of accuracy and in terms of what the floating point number looks like in binary and hex.