CS204 Data Representation/Built-in Types/Overflow - Lab Questions

1) Please make the decimal to binary conversion below.

120 :

2) Please make the binary to decimal conversion below.

0111 0001:

3) (a) Please convert the decimal numbers below to binary using Sign/Magnitude, 1's complement and 2's complement representations. Assume 8-bit storage.

i. Decimal number: 35
Sign/Magnitude Representation:
1's complement:
2's complement:

ii. Decimal number: -60
Sign/Magnitude Representation:
1's complement:
2's complement:

(b) Please convert the binary numbers below to decimal assuming that the binary numbers are represented in Sign/Magnitude, 1's complement and 2's complement.

iii. Binary number: 0110 0001
If this number is in Sign/Magnitude representation:
If this number is 1's complement:
If this number is 2's complement:

iv. Binary number: 1110 1101
If this number is Sign/Magnitude representation:
If this number is 1's complement:
If this number is 2's complement:

4) Please make the subtraction operation below after converting the operands to binary in Sing/Magnitude, 1's complement and 2's complement representations. Please do the math in binary.

110_{10} - 80_{10} = ?

5) What is the output?

a)
```
char ch;
ch = -190;
```

```
    cout << ch << endl;
    cout << (int)ch << endl;
```

b)
```
    char ch;
    ch = -67;

    cout << ch << endl;
    cout << (int) ch << endl;
```

c)
```
    unsigned char ch;
    ch = 200;

    cout << ch << endl;
    cout << (int) ch << endl;
```

d)
```
    unsigned char ch;
    ch = -67;

    cout << ch << endl;
    cout << (int) ch << endl;
```

e)
```
    short k = -61200;
    cout << "Short Integer: " << k << endl;
```

f)
```
    short ints = -20000;
    unsigned short intus = ints;
    cout << "implicit unsigned type-casting: " << intus << endl;
```

g)
```
    short  ints = -20000;
    cout << "explicit unsigned type-casting: " << (unsigned short) ints << endl;
```

h)
```
    unsigned short usnum = 25800;
    cout << "explicit signed type-casting: " << (short) usnum << endl;
```

i)
```
    unsigned short usnum2 = 68450;
    cout << "explicit signed type-casting: " << (short) usnum2 << endl;
```

6) What are the outputs of the following pieces of code?

a) Operations with different type of operands

```
int a = 5;
int b = -10;
unsigned int c = 3;

if (a+b-c < 0)
{
	cout << "The result should definitely be less than zero!" << endl;
	cout << "Expression: " << a+b-c << endl;
}
else
{
	cout << "Hmm, what is going on?" << endl;
	cout << "Expression: " << a+b-c << endl;
}
```

b) Overflow

```
unsigned char count = 13;
while(count > 0)
{
	cout << (int) count << endl;
	count = count-5;
}
```