HW2 选做

1. 可以用哈夫曼编码，因为它是可变长的，且考虑到出现概率越大的数编码越短，总编码bits期望更低，于是节约了空间，达到了数据传输及压缩存储的目的。

64位机器

2. 分别是 1，1，0，1，0，1，1，1

无论是补码表示还是无符号数表示，0都是全0，于是 0==0U；

正常的补码比较，显然 -1<0；

有符号的-1与无符号0比较，-1转为无符号数，真值最大，-1>0U；

2147483647是32位最大有符号数，补码是011…1，而-2147483647-1则是100…0是最小的有符号数.

是正常比较，于是 2147483647 > -2147483647-1；

-2147483647-1 有符号，但与无符号2147483647比较，转化为无符号，

而无符号规则中 100…0 > 011…1，于是 2147483647U < -2147483647-1；

int解析时

2147483648U 溢出，进入下一个循环，变为 -2147483648，于是 2147483647 > (int)2147483648U

正常比较，-1 > -2 显然；

-1补码 11…1，转为unsigned后比-2大，于是 (unsigned) -1 > -2

-2补码 11…10

16bit 4bit 12bit
11…11 1000 0…0

64位机器

3. 输出为：

	(十进制)	(十六进制)
si :	-32768	ffff8000
usi :	32768	8000
i :	-32768	ffff8000
ui :	32768	8000

short 仅 2 byte，范围 $-2^{15} \sim 2^{15}-1$，即 -32768 ~ 32767.

而 -32768 补码 1000 0…0（12 bit），十六进制解析后是 ffff8000；~~符号位自动填充到高字节~~

si 转为无符号数时，最高位失去符号意义，变为数据位，十进制变为 32768，高位置零，补码为 00…10…（32bit 31bit）~~高位自动填充0~~，十六进制解析后是 8000；

有符号数 si 被长类型解析，符号位填充到高字节，得 ffff8000；

~~被短类型解析，符号位从高位丢失，~~

无符号 usi 被长类型扩展，高位填0，于是是 8000，数值不变，