**2-1：**

理論上來說，32bit的最小正值的bit pattern為00000000100000000000000000000000，不全為0因為全為0的話輸出值就是0了，故最小正值為2^(-126)，經計算機運算與題目相等無誤。

但是以64bit的double來儲存浮點數的話，最小正值可為2^(-1023)，故在64bit時的最小值並不是2^(-126)，因此要確定題目中的1.175494350822287507968736537222245677818665556772087521508751706278417259454727172851560500000000000000000000000000000000e-38f

是否為最小正值，要視儲存的bit數而定。

**2-2：**

The bit pattern of f=0.0 is 00000000000000000000000000000000(in float)

0000000000000000000000000000000000000000000000000000000000000000 (in double)

**2-3：**

我做了簡易的code來驗證結果

Code:

#include <stdio.h>

int main()

{

float f2;

f2=1.175494350822287500e-38f;

float f1;

f1=1.175494350822287507968736537222245677818665556772087521508751706278417259454727172851560500000000000000000000000000000000e-38f;

if(f1==f2) printf("%0.100e\n=\n%0.100e",f1,f2);

else printf("%0.100e\n!=\n%0.100e",f1,f2);

return 0;

}

Compilation:

gcc -o hw6\_3 hw6\_3.c

Execution:

./hw6\_3

Output:

1.1754943508222875000000000000000000000000000000000000000000000000000000000000000000000000000000000000e-038

=

1.1754943508222875000000000000000000000000000000000000000000000000000000000000000000000000000000000000e-038

因為儲存的bit數不夠，系統只儲存到1.1754943508222875e-38f，後面的數都變為0，故f1==f2。