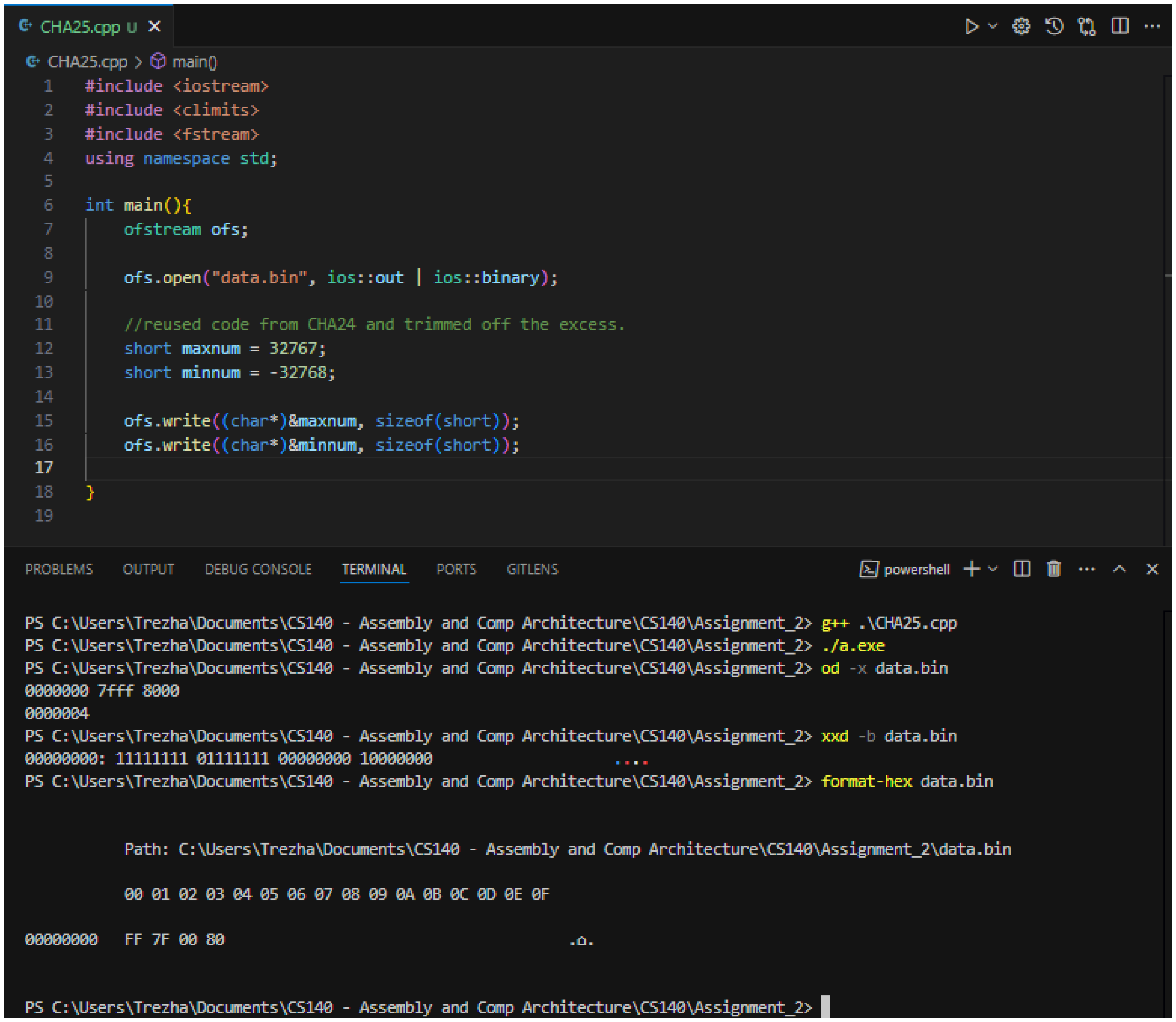
Assignment 2-5 Check integer value in the Binary file with C++ Program.



```cpp
#include <iostream>
#include <climits>
#include <fstream>
using namespace std;

int main(){
    ofstream ofs;

    ofs.open("data.bin", ios::out | ios::binary);

    //reused code from CHA24 and trimmed off the excess.
    short maxnum = 32767;
    short minnum = -32768;

    ofs.write((char*)&maxnum, sizeof(short));
    ofs.write((char*)&minnum, sizeof(short));

}

```

```
PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2> g++ .\CHA25.cpp
PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2> ./a.exe
PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2> od -x data.bin
0000000 7fff 8000
0000004
PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2> xxd -b data.bin
00000000: 11111111 01111111 00000000 10000000                 ....
PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2> format-hex data.bin


           Path: C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2\data.bin

           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   FF 7F 00 80                                      .⌂.

PS C:\Users\Trezha\Documents\CS140 - Assembly and Comp Architecture\CS140\Assignment_2>
```

Original:
32767 (0x7fff) (00007fff) (01111111(msb) 11111111)
-32767 (0x8000) (10000000(msb) 00000001)

We can see that the output of **xxd showcases the values in Little Endian**.
The msb for 32767 is located in the 2nd memory address instead of the first.
The msb for -32767 is located in the 4th memory address instead of the third.

Using **format-hex** also proves this as
It **prints out** 7fff **in**to ff7f and 8000 into 0080, which is also **Little Endian**.

The addresses in which they're saved also shows how the LSB takes priority in using up the lower address. IE. 00 = FF, 01 = 7F and 02 = 00, while 03 = 80.

In 2's complement system, the MSB serves as the sign bit. 0 for pos, 1 for neg.
For 32767
We can see how it corresponds to this format, with (0111 1111 1111 1111) leading with a 0, which means it's a positive value.
And for -32767
We can see how it also corresponds, with (1000 0000 0000 0000) leading with a 1 which shows it's negative, and complements 32767 perfectly. The value is represented by flipping all the bits of the absolute value and adding 1. So as we can observe in the binary file, due to the little-endian system, the least significant byte comes first, which explains why we see 32767 print as ff7f and -32768 print as 0080.