Write a C function with the prototype:

```c
void itobin(short int ival, char *buf)
```

that takes the short integer `ival` and produces a character string of one and zero characters contained in the string `buf`, representing the bits of `ival` in binary form.

For example, assuming a 16-bit short integer size,

- If `ival`=23, the binary string should be: “0000000000010111”
- If `ival`=-31000, the binary string (2’s complement) should be: “1000011011101000”

Your function will need to determine the size in bytes of a `short int` for the compiler you use.

**Question:** do you also need to consider the “endian” storage format of the processor?

The `main()` program that calls your function `itobin()` must declare the character string `buf` to be long enough to hold the required number of bits, e.g., if a `short int` is 4 bytes, then `buf` must be able to hold 32 characters.

**Question:** do you need to declare the total string length plus space for the null string terminator, or does the C compiler take care of that?

Write a `main()` routine that tests your `itobin()` function by allowing the user to enter an integer, then prints the integer in decimal and the string produced by `itobin()` including leading zeros.

Show your results for a comprehensive variety of positive and negative values of `ival`, and explain your testing method.