1 Exercise: Printing Bits

Write a C program that reads an integer from the keyboard, and prints out its bit pattern on the screen, as four eight-bit blocks. So, for example, if the user enters 1535, the computer should print:

```
00000000 00000000 00000101 11111111
```

Make sure this works for negative numbers. You should write a separate function that takes an integer argument, and prints it in the format shown above. Use a function prototype to declare the function, and place `main()` above the function.

2 Exercise: Bit Flipping

A trick sometimes used in programming is to use individual bits in an integer as one-bit flags. Write a C program that flips the value of the requested bit of an integer variable, and displays the variable on the screen. The program should continue asking for bit number and flipping bits until the user enters a negative number for the bit position. Here is what a sample run should look like:

```
00000000 00000000 00000000 00000000
Which bit shall I flip? 0
00000000 00000000 00000000 00000001
Which bit shall I flip? 7
00000000 00000000 00000000 10000001
Which bit shall I flip? 31
10000000 00000000 00000000 10000001
Which bit shall I flip? 7
10000000 00000000 00000000 00000001
Which bit shall I flip? -1
```

For this, you should make use of the function you have written in the previous exercise.

3 A Solution for Printing Bits

```c
#include <stdio.h>

void print_binary(int x);

int main(void)
{
  int a;

  printf("Enter a number:");
  scanf("%d", &a);

  print_binary(a);

  return 0;
}


void print_binary(int x)
{
  unsigned int mask;
  int bits_printed=0;

  for (mask = 1 << 31; mask > 0; mask >>= 1) {
    printf((x & mask) ? "1" : "0");

    bits_printed++;

    if (bits_printed == 8) {
      printf(" ");
      bits_printed = 0;
    }

  }

  printf("\n");

}
```