2. 정수의 비트표현, 비트단위 연산 결과 출력을 위한 함수 및 응용 프로그램 작성 (20점, 30분)

1) 정수 (integer) 자료형 인수를 전달받아 32 비트 형식으로 출력하는 함수 void printInt_Bits(int d)를 구현하라. 8비트 마다 한 칸을 뛰어 쉽게 구분할 수 있게 할 것.
2) 2개의 정수와 연산자를 인수로 전달받는 printIntOper_Bits(char *bop, int x, int y) 함수를 구현하라. 이 함수에서 지원되는 연산은 정수에 대한 덧셈, 뺄셈 연산과 비트 단위 AND (&), 비트단위 OR (|), SHIFT_LEFT(<<), SHIFT_RIGHT(>>)를 계산하고, printInt_Bits() 함수를 사용하여 다음과 같은 포맷으로 출력할 것.
3) main() 함수에서는 “127 + 5” 또는 “10 >> 3”과 같은 형식으로 “숫자 연산자 숫자“로 입력 받은 후, printIntOper_Bits() 함수를 호출하여 정수의 비트단위 연산 결과를 출력.

```
/* 주석문 */
. . .  // 필요한 전처리기 추가
void printInt_Bits(int d);
void printBinOper_Bits(char *bop, int a, int b);
void main()
{
        int a, b;
        char op[4] = "";
        while (1)
        {
                printf("Input integer operations in format int_a op int_b : ");
                scanf("%d %s %d", &a, op, &b);
                if (strcmp(op, ".") == 0)
                        break;
                printBinOper_Bits(op, a, b);
        }

}
. . .  // 필요한 함수 구현
```

4) 실행결과 (예시)

```
Input integer operations in format int_a op int_b : 127 + 5
127 + 5 => 132
   00000000 00000000 00000000 01111111
 + 00000000 00000000 00000000 00000101
 -----------------------------------
   00000000 00000000 00000000 10000100
Input integer operations in format int_a op int_b : 10 >> 3
10 >> 3 => 1
   00000000 00000000 00000000 00001010
>> 00000000 00000000 00000000 00000011
 -----------------------------------
   00000000 00000000 00000000 00000001
Input integer operations in format int_a op int_b : 10 << 5
10 << 5 => 320
   00000000 00000000 00000000 00001010
<< 00000000 00000000 00000000 00000101
 -----------------------------------
   00000000 00000000 00000001 01000000
Input integer operations in format int_a op int_b : 10 & 12
10 & 12 => 8
   00000000 00000000 00000000 00001010
 & 00000000 00000000 00000000 00001100
 -----------------------------------
   00000000 00000000 00000000 00001000
Input integer operations in format int_a op int_b : 10 | 12
10 | 12 => 14
   00000000 00000000 00000000 00001010
 | 00000000 00000000 00000000 00001100
 -----------------------------------
   00000000 00000000 00000000 00001110
Input integer operations in format int_a op int_b : 127 - 5
127 - 5 => 122
   00000000 00000000 00000000 01111111
 - 00000000 00000000 00000000 00000101
 -----------------------------------
   00000000 00000000 00000000 01111010
Input integer operations in format int_a op int_b :
```