2.60

```c
#include <stdio.h>
unsigned replace_byte(unsigned x, int i, unsigned char b) {
    unsigned tmp = (1 << 8) - 1;
    return x ^ (x & tmp << (i << 3)) ^ (b << (i << 3));
}
int main() {
    printf("%x\n", replace_byte(0x12345678, 2, 0xAB));
    printf("%x\n", replace_byte(0x12345678, 0, 0xAB));
}
```

2.61

```c
#include <assert.h>
#include <stdio.h>

int calcA(unsigned x) { return !!x; }
int calcB(unsigned x) { return !!(~x); }
int calcC(unsigned x) { return !!(x & (1 << 8) - 1); }
int calcD(unsigned x) {
    int w = sizeof(int) << 3;
    return !!(~(x >> (w - 8)));
}
int main() {
    unsigned x;
    scanf("%u", &x);
    int A = calcA(x);  // A : any bit of x equals 1
    int B = calcB(x);  // B : any bit of x euqals 0
    int C = calcC(x);  // C : any bit in the least significant byte of x equals 1
    int D = calcD(x);  // D : any bit in the most significant byte of x equal 0
    printf("%d %d %d %d\n", A, B, C, D);
}
```

2.65

```c
#include <assert.h>
#include <stdio.h>
// w=32
// return 1 when x contains an odd number of 1s;
int odd_ones(unsigned x) {
    x ^= (x >> 16);
    x ^= (x >> 8);
    x ^= (x >> 4);
    x ^= (x >> 2);
    x ^= (x >> 1);
    return x & 1;
}

int main() {
    // test
    for (unsigned i = -1;; i--) {
        if (i & 65536) printf("%u\n", i);
        assert(odd_ones(i) == (__builtin_popcount(i) & 1));
    }
}
```