### Setting a Bit (make it 1)

If we want to **set the k-th bit** of a number:

num = num | (1 << k);

Example:

```
#include <stdio.h>
int main() {
    int num = 5;   // 0101 in binary
    int k = 1;     // set bit at position 1 (0-based from right)
    num = num | (1 << k);
    printf("After setting bit %d: %d\n", k, num);
    return 0;
}
```

**W.A.P to set (make bit state = 1) the 4th and 7th position of a given number and print the result.**

( Assume positions are **0-based** from the rightmost bit).

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    // Set 4th and 7th bits (0-based indexing)
    num = num | (1 << 4);   // set 4th bit
    num = num | (1 << 7);   // set 7th bit

    printf("Number after setting 4th and 7th bits = %d\n", num);

    return 0;
}
```

Mask value:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    // Create mask for 4th and 7th bits
    int mask = (1 << 4) | (1 << 7);

    // Apply mask
    int result = num | mask;

    printf("Original number = %d\n", num);
    printf("Mask = %d\n", mask);
    printf("Number after setting 4th and 7th bits = %d\n", result);

    return 0;
```

```
}
```

### Clearing a Bit (make it 0)

If we want to **clear the k-th bit**:

- num = num & ~(1 << k);

### Toggling a Bit (flip 0→1 or 1→0)

If we want to **toggle the k-th bit**:

num = num ^ (1 << k);

### Testing if a Bit is Set

If we want to **check** whether the k-th bit is 1:

```
if (num & (1 << k)) {
    // bit is set
}
```