**2.** Given a positive integer n, count the total number of set bits in binary representation of all numbers from 1 to n.

#include <stdio.h>

unsigned int getLeftmostBit(int n)

{

int m = 0;

while (n > 1) {

n = n >> 1;

m++;

}

return m;

}

unsigned int getNextLeftmostBit(int n, int m)

{

unsigned int temp = 1 << m;

while (n < temp) {

temp = temp >> 1;

m--;

}

return m;

} unsigned int \_countSetBits(unsigned int n, int m);

unsigned int countSetBits(unsigned int n)

{

int m = getLeftmostBit(n);

return \_countSetBits(n, m);

}

unsigned int \_countSetBits(unsigned int n, int m)

{

if (n == 0)

return 0; m = getNextLeftmostBit(n, m);

if (n == ((unsigned int)1 << (m + 1)) - 1)

return (unsigned int)(m + 1) \* (1 << m);

n = n - (1 << m);

return (n + 1) + countSetBits(n) + m \* (1 << (m - 1));

}

int main()

{

int n = 3;

printf("Total set bit count is %d", countSetBits(n));

return 0;

}

**Examples:**  
Input: n = 3  
Output: 4  
Input: n = 6  
Output: 9

**Hint:** Read a positive integer (example: 3 indicates range), so u have to consider 1, 2, 3 as the input convert these numbers into binary and count the number of 1 in that (1- 0001, 2- 0010, 3- 0011) number of 1s from all 3 digit is 4 so the answer is 4



