|  |  |
| --- | --- |
| **Copy Set Bits in a range in C++** | |
| #include <iostream>  using namespace std;  int copySetBitsInRange(int a, int b, int left, int right) {  int m = (1 << (right - left + 1)) - 1; // Creates a mask of 1s of the required length  m = (m << (left - 1)); // Shifts the mask to the correct position  m = (m & a); // Extracts the bits from 'a' that need to be copied  b = b | m; // Copies the extracted bits to 'b'  return b; // Returns the result  }  int main() {  int a = 5;  int b = 3;  int left = 1;  int right = 1;  b = copySetBitsInRange(a, b, left, right);  cout << b << endl;  return 0;  } | int a = 5; // binary: 0101  int b = 3; // binary: 0011  int left = 1;  int right = 1;  We want to copy **only bit 1** (LSB) from a to b.  **🔍 Step-by-step Dry Run:**   | **Step** | **Expression** | **Result (in binary)** | **Explanation** | | --- | --- | --- | --- | | 1 | (1 << (right - left + 1)) - 1 | (1 << 1) - 1 = 1 → 0001 | Create a mask of 1s of length right - left + 1. | | 2 | m = m << (left - 1) → 1 << 0 = 1 | 0001 | Shift the mask to the correct bit position range (left to right). | | 3 | m = m & a → 0001 & 0101 = 0001 | 0001 | Mask a to extract the set bits in that range. | | 4 | `b = b | m→0011 | 0001 = 0011` | | 5 | return b | 3 | Final result. |   **🔚 Final Output:**  cout << b << endl; // 3  So the output is:  3  **✅ Summary Table**   | **Variable** | **Value (decimal)** | **Binary** | | --- | --- | --- | | a | 5 | 0101 | | b (before) | 3 | 0011 | | Mask | 1 | 0001 | | Masked a | 1 | 0001 | | b (after) | 3 | 0011 |   Nothing changed in b, because bit 1 was already set in both a and b. |
| 3 | |