

1) read two integers values perform bitwise operations

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

```
int main() {
    int num1, num2;
    int result_and, result_or, result_not_num1, result_not_num2;

    // Prompt the user to enter two integer values
    printf("Enter the first integer: ");
    scanf("%d", &num1);

    printf("Enter the second integer: ");
    scanf("%d", &num2);

    // Perform Bitwise AND operation
    result_and = num1 & num2;
    printf("\nBitwise AND of %d and %d is: %d\n", num1, num2, result_and);

    // Perform Bitwise OR operation
    result_or = num1 | num2;
    printf("Bitwise OR of %d and %d is: %d\n", num1, num2, result_or);

    // Perform Bitwise NOT operation on num1
    result_not_num1 = ~num1;
    printf("Bitwise NOT of %d is: %d\n", num1, result_not_num1);

    // Perform Bitwise NOT operation on num2
    result_not_num2 = ~num2;
    printf("Bitwise NOT of %d is: %d\n", num2, result_not_num2);

    return 0;
}
```

2) mcq on preincrement, predecrement, leftshift, rightshift

- Question: What is the output of the following C code snippet?

```
int x = 5;
int y = ++x;
printf("%d, %d", x, y);
```

a) 5, 5 b) 6, 5 c) 5, 6 d) 6, 6 Answer: d) 6, 6

- Question: If `int a = 10;` and `int b = --a;`, what are the values of `a` and `b` after these statements?

a) a = 10, b = 9 b) a = 9, b = 10 c) a = 9, b = 9 d) a = 10, b = 10 Answer: c) a = 9, b = 9

- Question: Which of the following statements correctly uses the pre-increment operator to increment i and assign the new value to j?

a) j = i++; b) j = ++i; c) j = i--; d) j = --i; Answer: b) j = ++i;

- Question: Consider `int p = 7; int q = 3; int r = ++p + --q;`. What is the value of r?

a) 10 b) 11 c) 9 d) 8 Answer: a) 10 (p becomes 8, q becomes 2; $8 + 2 = 10$) [1]

- Question: What is the key difference between pre-increment and post-increment operators?

a) Pre-increment performs the operation after the value is used, post-increment before. b) Pre-increment performs the operation before the value is used, post-increment after. c) They are identical in functionality. d) Pre-increment works with integers, post-increment with floats. Answer: b) Pre-increment performs the operation before the value is used, post-increment after.

Left Shift Operator (<<)

- Question: What is the result of `10 << 1`?

a) 5 b) 10 c) 20 d) 1 Answer: c) 20 (equivalent to $10 * 2^1$)

- Question: If `int num = 3;`, what is the value of `num << 2`?

a) 3 b) 6 c) 12 d) 9 Answer: c) 12 (equivalent to $3 * 2^2$)

- Question: The left shift operator << effectively performs:

a) Division by 2 b) Multiplication by 2 c) Modulo operation d) Subtraction Answer: b) Multiplication by 2

- Question: What happens to the bits when a number is left-shifted?

a) Bits are shifted to the right, and the leftmost bits are discarded. b) Bits are shifted to the left, and the rightmost bits are filled with zeros. c) Bits are swapped. d) No change occurs. Answer: b) Bits are shifted to the left, and the rightmost bits are filled with zeros.

- Question: What is the binary representation of `7 << 2`? (Assume 8-bit representation for 7 is 00000111)

a) 00000001 b) 00011100 c) 00000011 d) 00001110 Answer: b) 00011100 (7 becomes 28)
Right Shift Operator (>>)

- Question: What is the result of $20 \gg 1$?

a) 40 b) 20 c) 10 d) 1 Answer: c) 10 (equivalent to $20 / 2^1$)

- Question: If `int val = 15;`, what is the value of `val >> 2`?

a) 15 b) 7 c) 3 d) 4 Answer: c) 3 (integer division of 15 by 2^2 , which is $15 / 4 = 3$)

- Question: The right shift operator >> effectively performs:

a) Multiplication by 2 b) Addition c) Integer division by 2 d) Bitwise AND Answer: c) Integer division by 2

- Question: When a positive number is right-shifted, what happens to the leftmost bits?

a) They are filled with ones. b) They are filled with zeros. c) They are discarded. d) They remain unchanged. Answer: b) They are filled with zeros.

- Question: What is the output of `printf("%d", -8 >> 1)`; in a typical C environment?

a) -4 b) 4 c) -5 d) Undefined behavior Answer: a) -4 (For signed numbers, the behavior of right shift is implementation-defined for negative numbers, but typically it performs arithmetic right shift, preserving the sign bit.)