## Day 15: Operators and Expressions

### 1. Write a program to demonstrate the use of all arithmetic operators.

#include <stdio.h>  
  
int main() {  
 int a = 10, b = 5;  
 int sum, difference, product, quotient, remainder;  
  
 // Addition operator (+)  
 sum = a + b;  
 printf("a + b = %d\n", sum);  
  
 // Subtraction operator (-)  
 difference = a - b;  
 printf("a - b = %d\n", difference);  
  
 // Multiplication operator (\*)  
 product = a \* b;  
 printf("a \* b = %d\n", product);  
  
 // Division operator (/)  
 quotient = a / b; // Integer division  
 printf("a / b = %d\n", quotient);  
  
 // Modulo operator (%)  
 remainder = a % b;  
 printf("a %% b = %d\n", remainder); // Use %% to print a literal %  
  
 return 0;  
}

### 2. Write a program to demonstrate the use of increment/decrement operators.

#include <stdio.h>  
  
int main() {  
 int a = 10;  
 int b = 10;  
  
 printf("Initial value of a: %d\n", a);  
 printf("Initial value of b: %d\n", b);  
  
 // Post-increment (value used then incremented)  
 printf("Value of a++: %d\n", a++); // Prints 10, then a becomes 11  
 printf("Value of a after a++: %d\n", a); // Prints 11  
  
 // Pre-increment (value incremented then used)  
 printf("Value of ++a: %d\n", ++a); // a becomes 12, then prints 12  
  
 printf("\n");  
  
 // Post-decrement (value used then decremented)  
 printf("Value of b--: %d\n", b--); // Prints 10, then b becomes 9  
 printf("Value of b after b--: %d\n", b); // Prints 9  
  
 // Pre-decrement (value decremented then used)  
 printf("Value of --b: %d\n", --b); // b becomes 8, then prints 8  
  
 return 0;  
}

### 3. Write a program to demonstrate relational operators.

#include <stdio.h>  
  
int main() {  
 int a = 10, b = 20;  
  
 printf("a = %d, b = %d\n", a, b);  
  
 // Equal to (==)  
 printf("a == b: %d\n", a == b); // 0 (false)  
  
 // Not equal to (!=)  
 printf("a != b: %d\n", a != b); // 1 (true)  
  
 // Greater than (>)  
 printf("a > b: %d\n", a > b); // 0 (false)  
  
 // Less than (<)  
 printf("a < b: %d\n", a < b); // 1 (true)  
  
 // Greater than or equal to (>=)  
 printf("a >= b: %d\n", a >= b); // 0 (false)  
  
 // Less than or equal to (<=)  
 printf("a <= b: %d\n", a <= b); // 1 (true)  
  
 return 0;  
}

### 4. Write a program to demonstrate logical operators.

#include <stdio.h>  
  
int main() {  
 int a = 5, b = 10, c = 15;  
 int result;  
  
 // Logical AND (&&)  
 // Returns 1 if both operands are non-zero (true), 0 otherwise  
 result = (a < b) && (b < c); // (true) && (true) = true (1)  
 printf("(a < b) && (b < c) is %d\n", result);  
  
 result = (a > b) && (b < c); // (false) && (true) = false (0)  
 printf("(a > b) && (b < c) is %d\n", result);  
  
 // Logical OR (||)  
 // Returns 1 if at least one operand is non-zero (true), 0 otherwise  
 result = (a < b) || (b > c); // (true) || (false) = true (1)  
 printf("(a < b) || (b > c) is %d\n", result);  
  
 result = (a > b) || (b > c); // (false) || (false) = false (0)  
 printf("(a > b) || (b > c) is %d\n", result);  
  
 // Logical NOT (!)  
 // Reverses the logical state of its operand  
 result = !(a < b); // !(true) = false (0)  
 printf("!(a < b) is %d\n", result);  
  
 result = !(a > b); // !(false) = true (1)  
 printf("!(a > b) is %d\n", result);  
  
 return 0;  
}

### 5. Write a program to swap two numbers using a temporary variable.

#include <stdio.h>  
  
int main() {  
 int num1, num2, temp;  
  
 printf("Enter first number: ");  
 scanf("%d", &num1);  
 printf("Enter second number: ");  
 scanf("%d", &num2);  
  
 printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);  
  
 temp = num1;  
 num1 = num2;  
 num2 = temp;  
  
 printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);  
  
 return 0;  
}

### 6. Write a program to swap two numbers without using a temporary variable.

#include <stdio.h>  
  
int main() {  
 int num1, num2;  
  
 printf("Enter first number: ");  
 scanf("%d", &num1);  
 printf("Enter second number: ");  
 scanf("%d", &num2);  
  
 printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);  
  
 // Using arithmetic operators  
 num1 = num1 + num2; // num1 now holds the sum  
 num2 = num1 - num2; // num2 gets original num1 (sum - original num2)  
 num1 = num1 - num2; // num1 gets original num2 (sum - new num2 which is original num1)  
  
 printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);  
  
 return 0;  
}

### 7. Write a program to evaluate a given arithmetic expression.

This program will demonstrate evaluating a simple hardcoded expression. For user-input expressions, a more complex parser would be needed.

#include <stdio.h>  
  
int main() {  
 // Example expression: (10 + 5) \* 2 - 3 / 1  
 int result = (10 + 5) \* 2 - 3 / 1;  
  
 printf("Result of (10 + 5) \* 2 - 3 / 1 = %d\n", result);  
  
 // Another example  
 float a = 15.0, b = 4.0, c = 2.0;  
 float expr\_result = (a / b) + c \* 3;  
 printf("Result of (15.0 / 4.0) + 2.0 \* 3 = %.2f\n", expr\_result);  
  
 return 0;  
}

### 8. Write a program to demonstrate bitwise AND, OR, XOR.

#include <stdio.h>  
  
int main() {  
 int a = 12; // Binary: 0000 1100  
 int b = 25; // Binary: 0001 1001  
 int result;  
  
 printf("a = %d (Binary: 0000 1100)\n", a);  
 printf("b = %d (Binary: 0001 1001)\n", b);  
  
 // Bitwise AND (&)  
 // Sets each bit to 1 if both bits are 1  
 result = a & b; // Binary: 0000 1000 (Decimal: 8)  
 printf("a & b = %d\n", result);  
  
 // Bitwise OR (|)  
 // Sets each bit to 1 if at least one of the bits is 1  
 result = a | b; // Binary: 0001 1101 (Decimal: 29)  
 printf("a | b = %d\n", result);  
  
 // Bitwise XOR (^)  
 // Sets each bit to 1 if only one of the bits is 1  
 result = a ^ b; // Binary: 0001 0101 (Decimal: 21)  
 printf("a ^ b = %d\n", result);  
  
 // Bitwise NOT (~) - Unary operator  
 // Inverts all the bits (0 becomes 1, 1 becomes 0)  
 // Note: Result depends on integer representation (2's complement for negative numbers)  
 result = ~a; // Binary: 1111 0011 (Decimal: -13, for 8-bit signed int)  
 printf("~a = %d\n", result);  
  
 // Left Shift (<<)  
 // Shifts bits to the left, filling with 0s on the right  
 result = a << 2; // Binary: 0011 0000 (Decimal: 48)  
 printf("a << 2 = %d\n", result);  
  
 // Right Shift (>>)  
 // Shifts bits to the right, filling with 0s on the left for unsigned,  
 // or sign bit for signed (arithmetic shift)  
 result = b >> 2; // Binary: 0000 0110 (Decimal: 6)  
 printf("b >> 2 = %d\n", result);  
  
 return 0;  
}

### 9. Write a program to find the largest of three numbers using conditional operator.

#include <stdio.h>  
  
int main() {  
 int num1, num2, num3, largest;  
  
 printf("Enter three numbers: ");  
 scanf("%d %d %d", &num1, &num2, &num3);  
  
 largest = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);  
  
 printf("The largest number is: %d\n", largest);  
  
 return 0;  
}

### 10. Write a program to check if a number is positive, negative, or zero.

#include <stdio.h>  
  
int main() {  
 int num;  
  
 printf("Enter a number: ");  
 scanf("%d", &num);  
  
 if (num > 0) {  
 printf("%d is a positive number.\n", num);  
 } else if (num < 0) {  
 printf("%d is a negative number.\n", num);  
 } else {  
 printf("%d is zero.\n", num);  
 }  
  
 return 0;  
}