**Program 925b**

**(Bitwise Operations)**

**Program Description:**

This assignment is designed to help students understand and work with bitwise operators. The goal is to apply bitwise operations such as AND, OR, XOR, NOT, and shifts on integers to achieve specific results.

* You are required to perform a series of operations using bitwise operators.
* You must not use loops or conditional statements; the focus is purely on bitwise manipulation.

### **Part 1: Understanding Basic Bitwise Operations**

Given two integers, a = 29 and b = 15, perform the following operations:

1. Bitwise AND: Perform a bitwise AND on a and b.
   * Expected result: a & b
2. Bitwise OR: Perform a bitwise OR on a and b.
   * Expected result: a | b
3. Bitwise XOR: Perform a bitwise XOR on a and b.
   * Expected result: a ^ b
4. Bitwise NOT: Perform a bitwise NOT on a. Note that depending on the language, you may need to limit the result to 8 or 16 bits.
   * Expected result: ~a
5. Left Shift: Shift the bits of a two positions to the left.
   * Expected result: a << 2
6. Right Shift: Shift the bits of b two positions to the right.
   * Expected result: b >> 2

### 

### **Part 2: Real-World Application with Bitwise Masks**

In this part, you will work with bitwise masks, which are often used to manipulate specific bits in data.

Given an 8-bit integer x = 0b11001010 (binary representation of 202), perform the following operations:

1. Clear a Bit: Clear the 4th bit (counting from 0) of x.
   * Hint: You can use the AND operator with a mask that has a 0 in the 4th bit position.
   * Expected result: x with the 4th bit cleared
2. Set a Bit: Set the 2nd bit of x.
   * Hint: Use the OR operator with a mask that has a 1 in the 2nd bit position.
   * Expected result: x with the 2nd bit set
3. Toggle a Bit: Toggle the 6th bit of x.
   * Hint: Use the XOR operator with a mask that has a 1 in the 6th bit position.
   * Expected result: x with the 6th bit toggled
4. Extract a Subset of Bits: Extract bits 2 to 4 (inclusive) from x. You can do this by shifting the bits and using a mask.
   * Expected result: The value of bits 2 to 4 extracted from x

### **Part 3: Combining Bitwise Operations for Complex Tasks**

1. Packing Two Values: Given two integers, m = 5 and n = 9, pack these two integers into a single 16-bit integer where m occupies the lower 8 bits and n occupies the upper 8 bits.
   * Expected result: A single 16-bit integer with m and n packed
2. Unpacking Values: Unpack the two values from the packed integer created in the previous step.
   * Expected result: The original values of m and n
3. Given a number y = 75, swap the 1st and 4th bits of y.
   * Hint: You can do this by extracting both bits, comparing their values, and then conditionally flipping them.
   * Expected result: y with its 1st and 4th bits swapped

**Statements Required:** Bitwise operators

**Expected Output:**

* Part 1:
  + a & b = 13
  + a | b = 31
  + a ^ b = 18
  + ~a = -30 (assuming 32-bit integers; adjust for 8-bit or 16-bit systems)
  + a << 2 = 116
  + b >> 2 = 3
* Part 2:
  + Clearing the 4th bit of x: 0b11000010 (194)
  + Setting the 2nd bit of x: 0b11001110 (206)
  + Toggling the 6th bit of x: 0b10001010 (138)
  + Extracting bits 2 to 4 from x: 0b001 (1)
* Part 3:
  + Packing m and n: 2309 (assuming m = 5 and n = 9)
  + Unpacking should give back m = 5 and n = 9.
  + Swapping the 1st and 4th bits of y = 75 should result in 67.