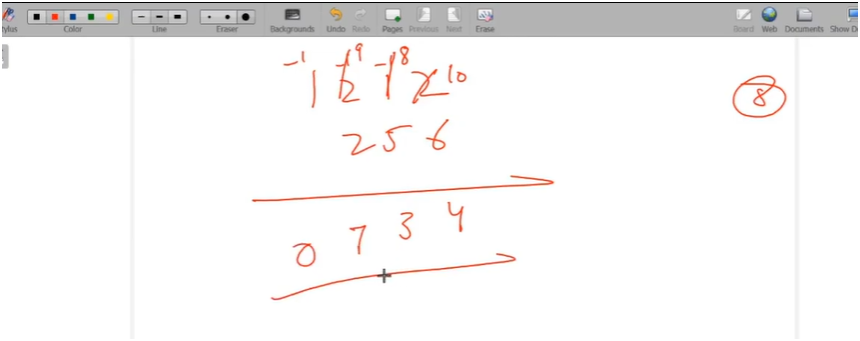
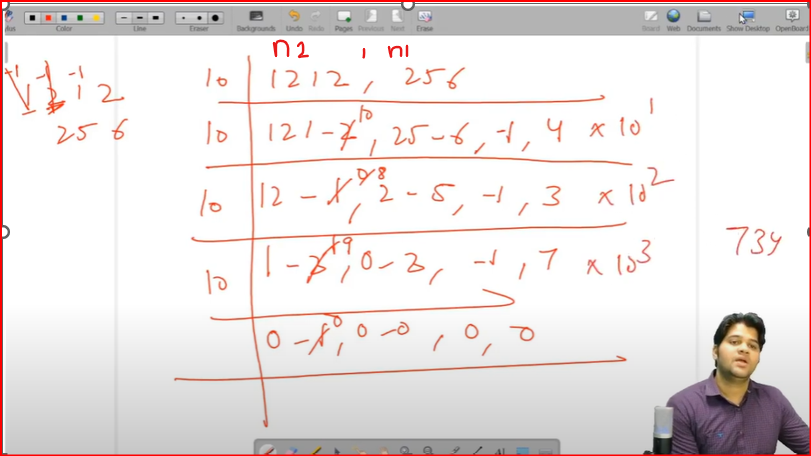


OCTA 8 TIMES OF EACH PLACE IF NOT SUBTRACT BORROW -1 AND ADD BASE(8)





**IF U OBSERVE CLEARLY THERE IS A PATTERN OF NUMBER 2,NUMBER 1|BORROW|SUBTRACTION OF REMAINDER**

* WHILE LOOP RUN UNTIL LARGEST NUMBER DOESN’T GETS EQL TO ZERO
* THEN GET REMAINDER FOR TWO NUMBERS BY DOING EACH NUMBER MODULUS BY 10
* AND THEN GET QUOTIENT FOR TWO NUMBERS BY DOING DIVIDE TO REDUCE THE NUMBER
* THEN CREATE A DEMO VARIABLE D AND ASSIGNED ZERO TO IT
* THEN ADD PREVIOUS BORROW TO RAMAINDER 2 OF DIGIT2 MODULUS 10
* THEN CHECK WHETER REMAINDER OF LARGER NUMBER IS GREATER THAN OR EQL TO REAMAINDER OF SMALLER NUMBER
  + IF SO BORROW MUST BE ZERO THEN SUBTRACT REMAINDER 2 AND REMAINDER 1
    - REMAINDER2-REMAINDER1
* IF REAMAINDER OF LARGER NUMBER IS SMALLER THAN OR EQL TO REAMAINDER OF SMALLER NUMBER
  + THEN BORROW MUST BE -1 AND ADD BASE AND SUBTRACT REMAINDER 2 AND REMAINDER 1
    - REMAINDER\_2+BASE-REMAINDER\_1

**NOTE:-BUT BEFORE IT ADD PREVIOUS BORROW TO REMAINDER 2**

* THEN MULTIPLY THE SUBTRACTION OF REMAINDER WITH INCREASING POWER OF 10

**TRICK TO REMMEBER:-**

**IN BASE ADDITION**

* + ADD THE TWO REMAINDERS AND ADD PREVIOUS CARRY IN IT
  + THEN FIND CARRY AND LAST DIGIT BY DIVIDE AND MODULUS BY 1

**IN BASE SUBTRACTION**

* + ADD PREVIOUS BORROW TO GREATER NUMBERS REMAINDER
  + CHECK FOR BOTH REMAINDERS
    - IF LARGER NUMBER REAMINDER IS GREATER THAN SMALLLER NUMBER REMAINDER
      * THEN MAKE BORROW TO ZERO AND SUBTRACT BOTH REMAINDERS
    - IF NOT SO THEN
      * THEN MAKE BORROW TO -1 ADD BASE TO REMAINDER2 AND SUBTRACT IT FROM REAMINDER 1

ALGORITHM:-

**✅ Topic: Subtract Two Numbers in the Same Base (With Borrow)**

**🧠 Purpose:**

To perform subtraction of two numbers digit-wise, in any given **base** (e.g., base 8, base 2), **handling borrows** properly — similar to how we subtract by hand.

**🔁 Step-by-Step Logic:**

1. **Initialize:**
   * borrow = 0: used when a digit in number2 is smaller than corresponding digit in number1.
   * sum = 0: holds the final answer (constructed in reverse).
   * power = 1: used to build the result in the correct order (units, tens, hundreds...).
2. **Loop:**  
   Run while(number2 > 0)  
   This assumes number2 is the larger number and stops when it's fully processed.
3. **Inside the Loop:**
   * Extract last digits:

int digit2 = number2 % 10;

int digit1 = number1 % 10;

 Apply existing borrow to digit2.

 If digit2 >= digit1, no new borrow needed.

 Else, **borrow 1 from next higher digit** of number2, and add base to digit2.

1. Calculate the difference:
   * int d = digit2 - digit1;

5. **Build the result:**

* Multiply d by the current power (to put the digit in correct place)
* Increment power \*= 10 for next digit.

6. Update number1 and number2 (remove last digits):

number1 /= 10;

number2 /= 10;

**🧪 Example: Subtracting 256 - 123 in base 8**

* Let’s subtract 123 (octal) from 256 (octal)
* Apply base logic, borrow when needed
* Final result is calculated in reverse and printed

**✅ TIP TO REMEMBER:**

**When digit2 < digit1, borrow 1 from left (means add base to digit2 and set borrow = -1)**  
**Always keep track of power to maintain correct positional value**