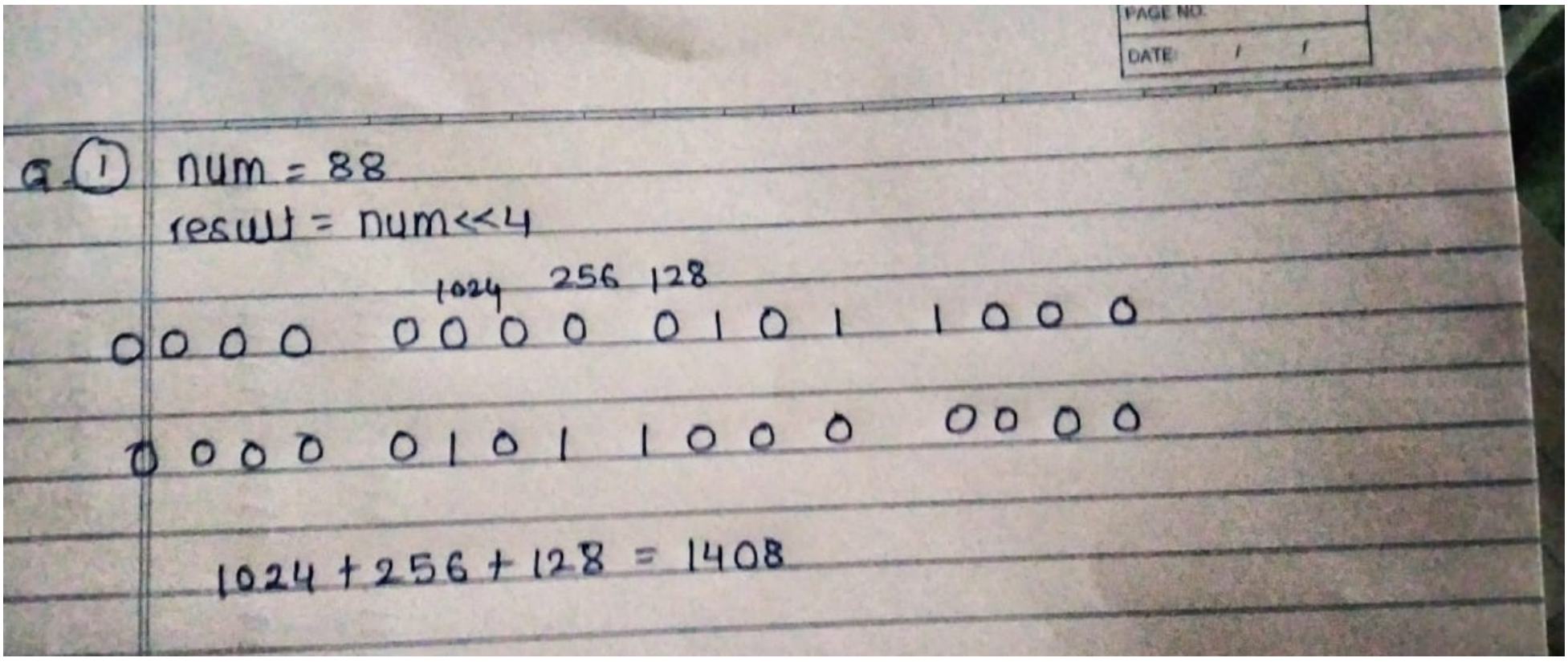
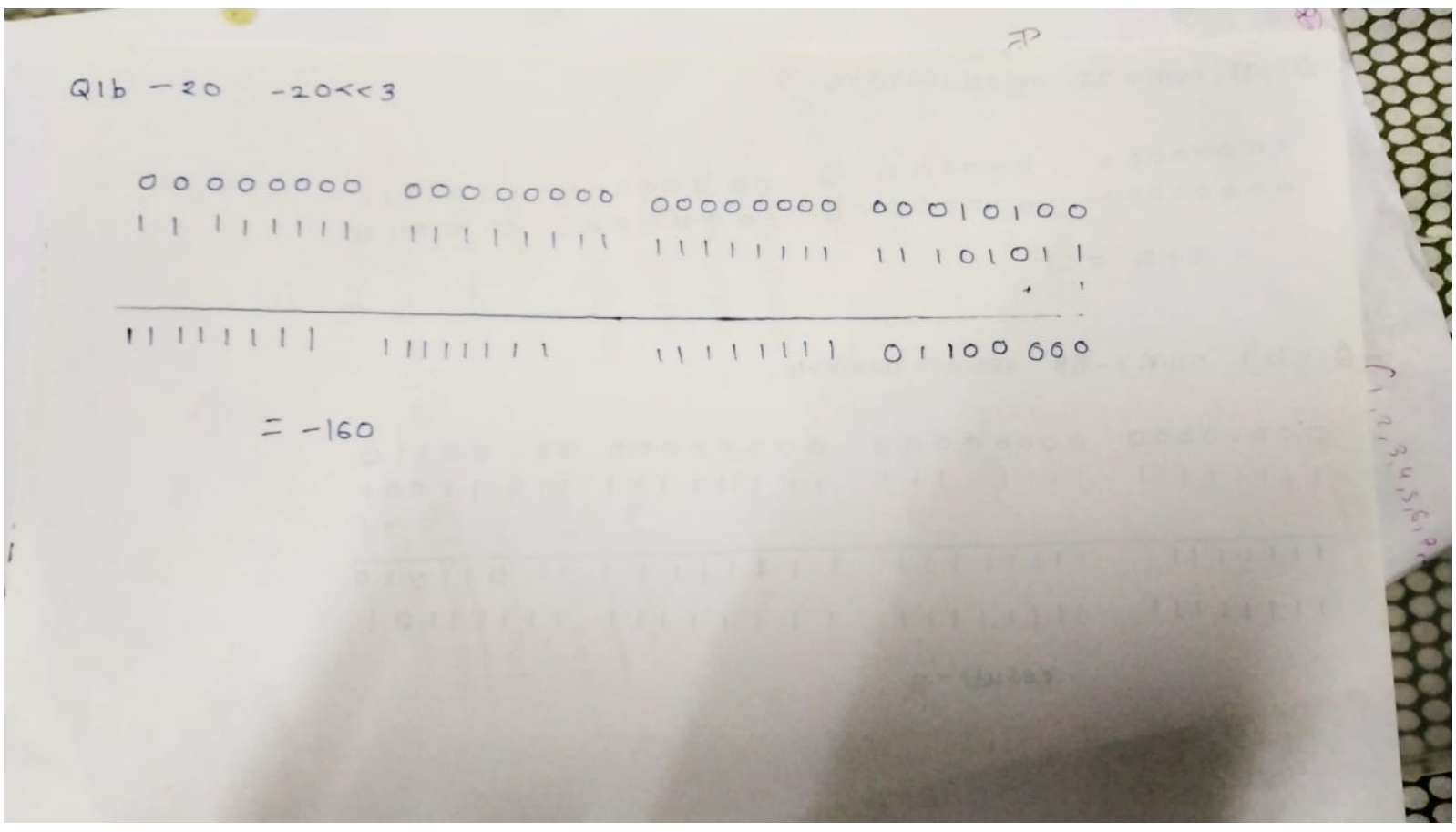
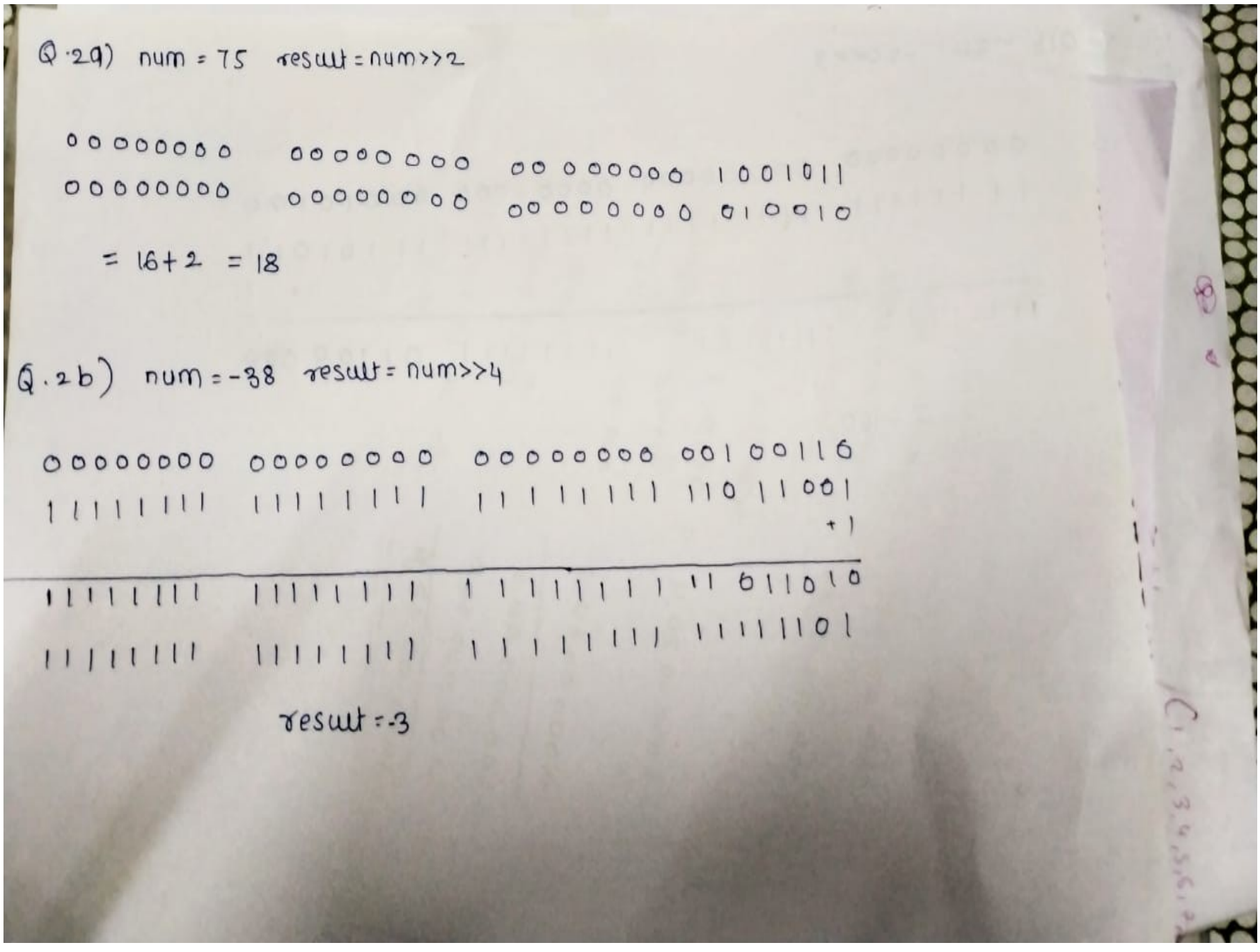
Q1b.


PAGE NO.
DATE: / /
a (1) num = 88
result = num<<4
1024 256 128
0000 0000 0101 1000
0000 0101 1000 0000
1024 + 256 + 128 = 1408

Q1b


Q1b −20 -20<<3
00000000 00000000 00000000 00010100
11111111 11111111 11111111 11101011
11111111 11111111 11111111 01100000
= −160

Q.2



Q3

```
class UnsignedShift{
    public static void main(String[] args) {
        int num=188, num1=255;
        int result = num>>>4;
        //int result1 = num1<<<2;
```

```
        int num2 = -108,num3=-123;
        int result2 = num2>>>23;
        //int result3 = num3<<<15;
    }
}
```

/*Explaination:
In the case of right shift Operator there is different behaviour for positive or negative values due to the signed bit to overcome this we have unsigned Right shift Operator.

But in case of left shift Operator there is no difference for positive or Negative values as the signed bits are not added by default but are the results of the Considered number hence there is no need of Unsigned Left shift Operator*/

Q.4
int a = 25, b = -34, c = 19, d = 4;
Expression 1 int res = ((a << 2 | b >> 3))
Expression 2 boolean result = ((a << 2 | b >> 3) < -10) || ((c++ << 2 & b-- >>> d) >= 23)

1.

Binary of a(25) = 0000 0000 0000 0000 0000 0000 0001 1001
a << 2 0000 0000 0000 0000 0000 0000 0110 0100 (Binary of 100)

Binary of b(-34) = 1111 1111 1111 1111 1111 1111 1101 1110 (Using 2's Complement)
b >> 3 1111 1111 1111 1111 1111 1111 1111 1011 (Binary of -5)

a << 2 0000 0000 0000 0000 0000 0000 0110 0100
|
b >> 3 1111 1111 1111 1111 1111 1111 1111 1011

1111 1111 1111 1111 1111 1111 1111 1111 (Binary of -1)

Find the Decimal Number from binary for Negative Numbers (Use exactly reverse for converting negative decimal to Binary)

1111 1111 1111 1111 1111 1111 1111 1111
- 1 Subtract 1

1111 1111 1111 1111 1111 1111 1111 1110 (Rule for binary Subtraction)

~ 0000 0000 0000 0000 0000 0000 0000 0001 (Complement the obtained Number)

Convert this to Decimal with negative sign i.e -1

Hence res = a << 2 | b >> 3 = -1

2. From previous Question we know a << 2 | b >> 3 = -1

Hence -1 < -10 || ((c++ << 2 & b-- >>> d) >= 23)

-1 > -10 (These are Negative) hence first part is false

c++ << 2 (post increment)

Binary of c++(19) = 0000 0000 0000 0000 0000 0000 0001 0011
19 << 2 = 0000 0000 0000 0000 0000 0000 0100 1100 (Binary of 76)

b-- >>> d (post decrement) and unsigned right shift

Binary of b--(-34) = 1111 1111 1111 1111 1111 1111 1101 1110
-34 >>> 4 = 0000 1111 1111 1111 1111 1111 1111 1101 (Binary of 268435453)

(No need for Conversions as there are operations are yet to complete)

19 << 2 0000 0000 0000 0000 0000 0000 0100 1100
&
-34 >>> 2 0000 1111 1111 1111 1111 1111 1111 1101

0000 0000 0000 0000 0000 0000 0100 1100 (Binary of 76)

76 >= 23 is true

Hence the overall expression is true (Logical OR operator)

result = true

Q5.

```
class ArithmeticOperations{

    static int a = 20, b = 15;

    static void binaryOps(){

        System.out.println("Static method of ArithmeticOperations class");
        System.out.println("Addition = "+(a+b));
        System.out.println("Subtraction = "+(a-b));
        System.out.println("Division = "+(a/b));
        System.out.println("Multiplication = "+(a*b));
        System.out.println();
    }

    void unaryOps(){

        System.out.println("Non-Static method of ArithmeticOperations class");
        System.out.println("Post Increment on a = "+(a++));
        System.out.println("Pre Increment on a = "+(++a));
        System.out.println("Post Decrement on b = "+(b--));
        System.out.println("Pre Decrement on b = "+(--b));
        System.out.println();
    }
}

class RelationalOperations{

    static int a = 25, b = 5;


    void intOperands(){

        System.out.println("Non-Static method of RelationalOperations class");
        System.out.println("Less Than Operator "+ (a < b));
        System.out.println("Greater Than Operator "+ (a > b));
```

```
        System.out.println("Equality Operator "+ (a == b));
        System.out.println("Not Equal Operator "+ (a != b));
        System.out.println("Greater Than Equal to Operator "+ (a >= b));
        System.out.println("Less Than or Equal to Operator "+ (a <= b));
        System.out.println();
}

static void boolOperands(){

        System.out.println("Static method of RelationalOperations class");
        System.out.println("Logical AND "+((a >= a++) && (b > --b )));
        System.out.println("Logical OR "+((a < ++a) || (b > --b )));
        System.out.println("Value of a after operations "+ a);
            System.out.println("Value of b after operations "+ b);
            System.out.println();

    }
}

class BitwiseOperations{

    static int a = 10, b = 20;

    static void basicOps(){

        System.out.println("Static method of RelationalOperations class");
        System.out.println("Bitwise And Operator " + (a & b));
        System.out.println("Bitwise Or Operator " + (a | b));
        System.out.println("Bitwise Xor Operator " + (a^b));
        System.out.println("Complement Operator " + (~b));
        System.out.println();
    }

    void shiftOps(){

        System.out.println("Non-Static method of RelationalOperations
class");
        System.out.println("Left Shift " + (a << 3));
        System.out.println("Signed Right Shift " + (b >> 4));
        System.out.println("Unsigned Right Shift " + (b >>> 4));
        System.out.println();
    }
```

```
}

class Main{

    public static void main(String[] args){

        ArithmeticOperations a = new ArithmeticOperations();
        ArithmeticOperations.binaryOps();
        a.unaryOps();

        RelationalOperations r = new RelationalOperations();
        RelationalOperations.boolOperands();
        r.intOperands();

        BitwiseOperations b = new BitwiseOperations();
        BitwiseOperations.basicOps();
        b.shiftOps();
    }
}
```