```
public class Cast{
    public static void main(String args[]){

        int i=10;

        long l=1;

        double d=3.4;

        byte b;
        b= (byte)i;
        System.out.println(b);

        i= (int) l;
        System.out.println(i);

        l=(long) d;
        System.out.println(l);
    }

}
```

b: 10

i : 10

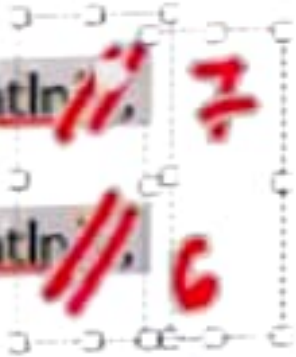
d : 3

CODING P •••

AT AB C0 N

ajela Tripa... Anandita Bh... CODING P

```
class Demo {

    public static void main(String args[]){

        int i=6;

        int j=i++;

        System.out.println(i); // 7

        System.out.println(j); // 6

    }
}
```



```
class Demo {

public static void main(String args[]) {

    byte i;
    int j=9;

    i= j;

 System.out.println(i );
}}
```

```
class Demo {

public static void main(String args[]) {

Boolean x= false;

System.out.println(- - - ); // !x
}}
```

```
class Demo {

public static void main(String args[]) {

    int i = -12;
    int j=13;

    i= i & j; //4

    i= i | j; // -3

    i= (byte) i;


 System.out.println(i );
}}
```

-12 Binary

0···0 | 0···0 | 0···0 | 0000 1100 (12)

1···1 | 1···1 | 1···1 | 1111 0100 (-12)

0···0 | 0···0 | 0···0 | 0000 1101 (13)

OR

1···1 | 1···1 | 1···1 | 1111 1101 (

=

2's complement

10

+1

11 (-3)

o/p :

```
class Demo {

public static void main(String args[]) {

    byte i;
    int j=9;
    yte z=10;

    i= j; // error : casting
    i= (byte) j;
 System.out.println(i );
}}
```

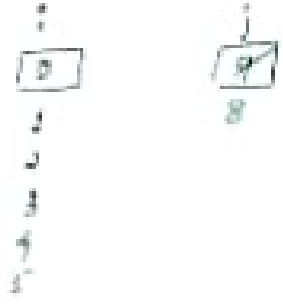
```
class Demo {

public static void main(String args[]) {
    int i=9;
    Int k = 0;


    If((++i==1 ||++k==1))
       System.out.println(i); //10
    System.out.println(k);// 1
 }
}
```

```
class Demo {

public static void main(String args[]) {
    int i=64;
    int k = 0;
    k= (i>k)? 10 : --;
    k= (i>k)? 10 : --;

 System.out.println(k); //10
}}
```

CODING P ...

```
class Demo {

public static void main(String args[]) {

    int i = 5;
    int j=i>>2;



 System.out.println(j ); //1
}}
```

Lamp V.1



(3<7)

(x==3) T

(3==4

3 (x<7)

(4<7) T

(4==3) F

or (x==5)

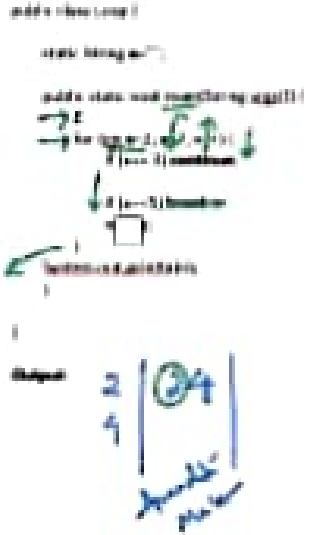
(4==5 F

5 (x<7)

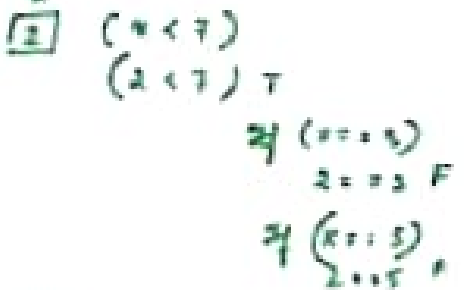
(5<7) T

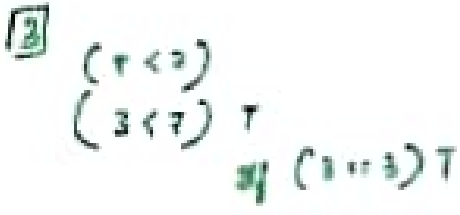
or (x==5)
5==3 F

or (x==5)
(5==5) T



String Concatenation -





Q 5. Enter a number and check whether it is a binary number or not

[illegible]

1020101 0 % 10 ;

Scan(S) = ;

int x = S.nextInt();

/ 10 ;

Assign V_1

11.) Given a number and check whether it is an Armstrong number or not

```
import java.io.*;

public class Armstrong {

    public static void main(String args[]) throws IOException

        int n, r=0,s=0,t;
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        t = n;

        while(t>0){
            r = t%10;
            s = s + (r*r*r);
            t = t/10;
        }
        if(n==s)
            System.out.println("Armstrong Number is:" + n);
        else
            System.out.println("Not an Armstrong Number" +n);

    }
}
```

Output

Assign V.S

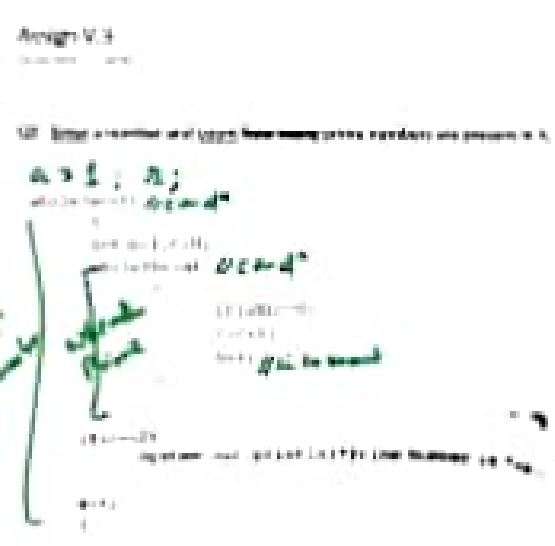
Q.9

Q.9 Draw the following structure

Q.10

Solution : PASCAL Pyramid

Assign V.1



```
{
public class Loop1
{
static int i; // 5 lab's

public static void main(String args[])
{
for(i=1;i<3;i++)
System.out.println(i);
int i; // local variable ←
for(i=1;i<=3;i++)
System.out.println(i);
System.out.println(Loop1.i);
}
}
```

error → duplicate variable

```
int i;
for ( i = 1; i++<3 ; ){
    S.o.p ( i );
```

1
2
2
3
3

$= \frac{(i-0)!(t-0+1)}{(i-0)!\,0!}$

$t : 1$

$\frac{t!}{(t-i)!}$

term j

term j-1

$\frac{{}^{n}C_{n}}{{}^{n}C_{n-1}} = \frac{\frac{n!}{n!\,(n-n)!}}{\frac{n!}{(n-0)!\,(n-n+1)!}}$

$= \frac{(n-1)!}{n!} \cdot \frac{(n-n+1)!}{(n-n)!}$

$= \frac{(n-n+1)}{n}$

$\frac{{}^{n}C_{n}}{{}^{n}C_{n-1}} = \frac{n-n+1}{n}$

$\Rightarrow {}^{n}C_{n} = {}^{n}C_{n-1} \cdot \frac{n-n+1}{n}$

CODING P ...

AB

SG

C0

N

handita Bh...

Sourav Gos...

CODING P

Assign V.1)

Q.9

Q.9 Draw the following structure

```
n

for (i=0; i<n ; i++)
    for(j=0; j<n-1 ; j++)
        {
          S.o.p(" ");
        }
    for(k=0; k<=i ; k++){


    }
}
```

Q.10

Structure : Pascal Pyramid / Triangle

Exercise V_2

```
2
public class Exp2
{
  public static void main(String args[])
    {
      int i=0;
      do
        {for(int i=0;i<=2;i++)
        {System.out.println("*");
      while(i<=2);
        }
    }
```

%

1
2
1
2
1
2

V.4

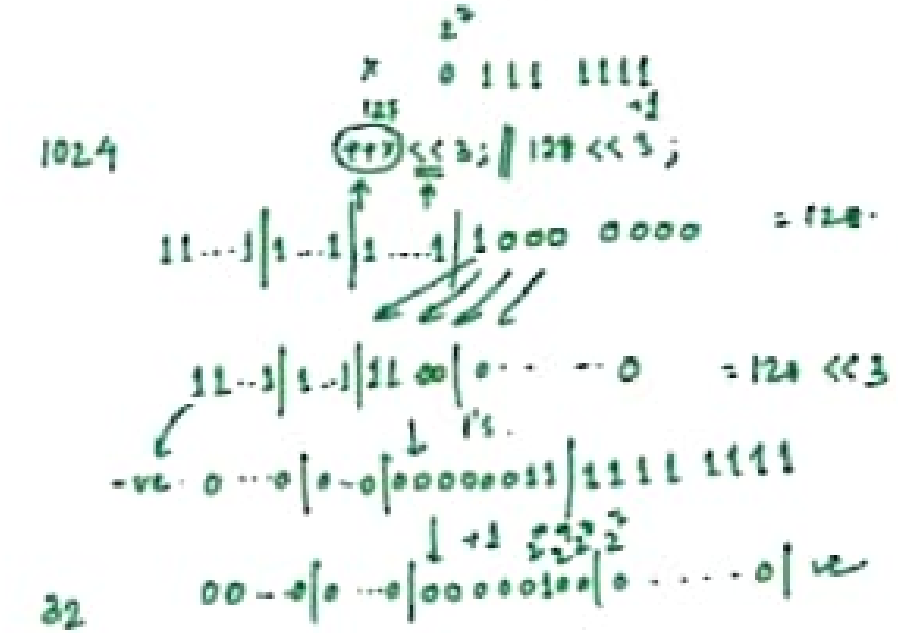
```
public class Nest2 {

    public static void main(String args[]) {
        int j=0;
        do
            for (int i=0;i++ <2; )
                System.out.println(i);
        while (j++<2)
    }
}
```

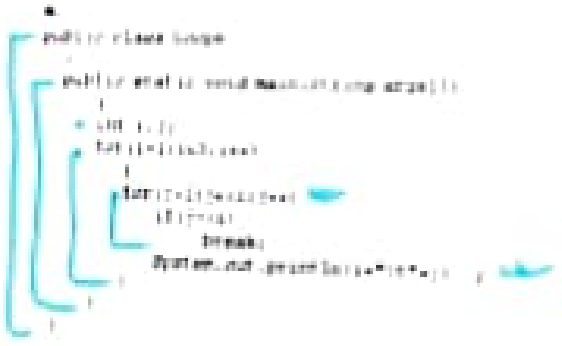
Output:

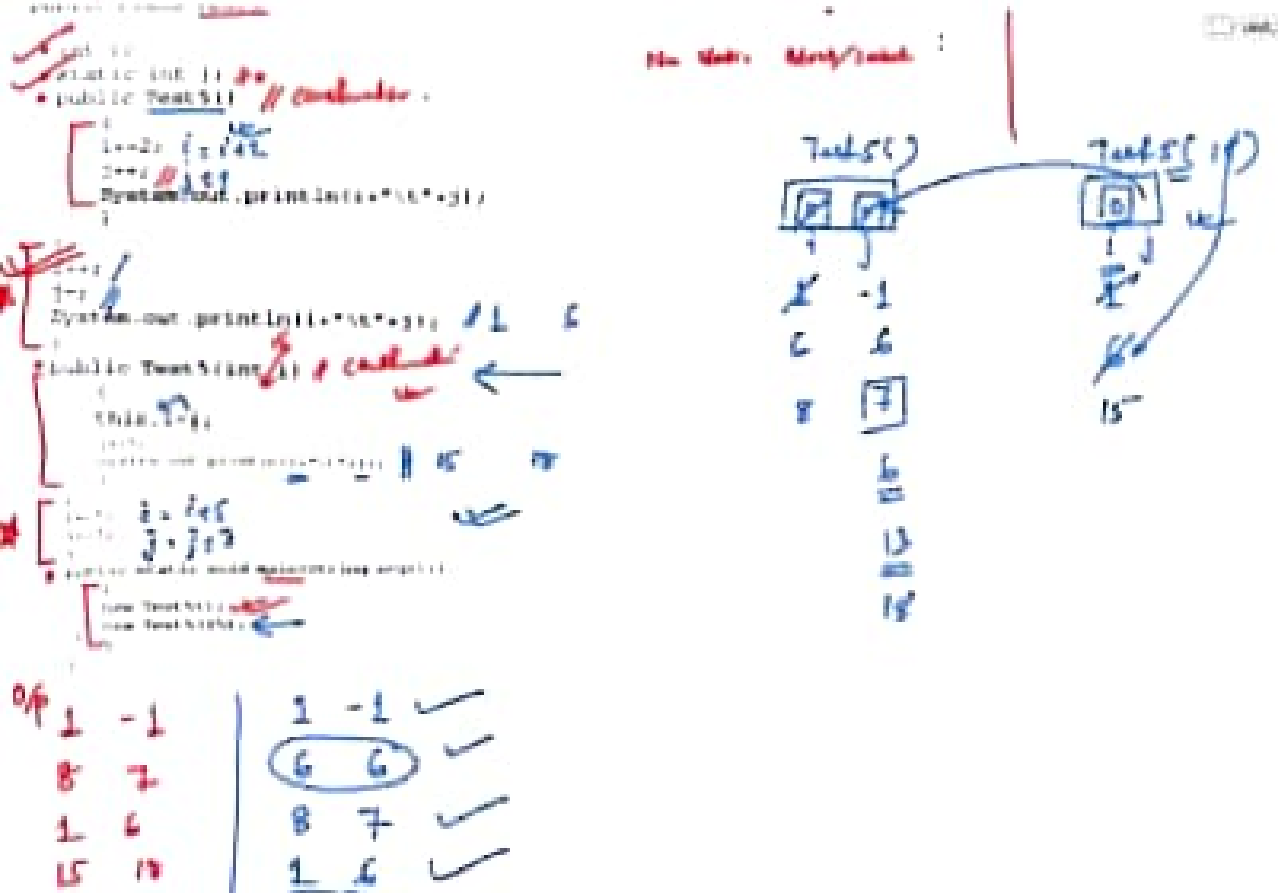
IV.9 TODO from this onwards

```
class Demo
{
    public static void main(String [] args)
    {
        int a=127;
        a=(a--)-3;
        System.out.println(a);
    }
}
```

```
class Demo
{
    public static void main(String [] args)
    {
        int a=127;
        a=a--+a++-3;
        System.out.println(a);
    }
}
```







```
class Demo
{
    public static void main(String [] args)
    {
        int a=127, b=128;
        System.out.println(a+b);
    }
}
```

5/8 508

```
class Demo
{
    public static void main(String [] args)
    {
        int a=127, b=128;
        a=(byte)a;
        System.out.println(a+b);
    }
}
```

8/8 128

```
class Demo
{
    public static void main(String [] args)
    {
        int a=127, b=128;
        a=a-1;
        System.out.println(a+b);
    }
}
```

8/8 128

```
class Demo
{
    public static void main(String [] args)
    {
        int a=127, b=128;
        a=(byte)a;
        System.out.println(a+b);
    }
}
```

8/8 255

Jara's

Precedence	Operator	Type	Associativity
17	[]	Parentheses Array subscript Member selection	Left to Right
16	++ --	Unary post-increment Unary post-decrement	Right to left
15	++ -- + - ! ~ (type)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
13	* / %	Multiplication Division Modulus	Left to right
12	+ -	Addition Subtraction	Left to right
11	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left
	&= ^= |= <<= >>= >>>=	Bitwise AND assignment Bitwise exclusive OR assignment Bitwise inclusive OR assignment Bitwise left shift assignment Bitwise right shift with sign extension assignment Bitwise right shift with zero extension assignment	

O/p 508

O/p 131

O/p 128

O/p 255

= 127 << (129) + (128)

= 127 << (257)

= 127 << (257 % 32)

= (127 << 2) =

= 127 * 2² = 127 * 4 = 508

Java's

Precedence	Operator	Type	Associativity
15			Left to Right
14			Right to left
13			Right to left
12			Left to right
11			Left to right
10			Left to right
9			Left to right
8			Left to right
7			Left to right
6			Left to right
5			Left to right
4			Left to right
3			Left to right
2			Right to left
1			Right to left

CODING P •••

BD

AB

C0

N

...kram Deo ...

Anandita Bh...

CODING P

```
}
```

```
class Demo {

public static void main(String args[]) {

 int x = 12;
Int y =5;
x=%y;
 System.out.println(x );
}}
```

% :

```
class Demo {
    public static void main(String [ ] args){
        int x=12;
        int y=-x;
        System.out.print (y);
    }
}
```

Output : -12

```
class Demo {

public static void main(String args[]) {

 int x = 12;
Int y =5;
x%=y;
 System.out.println(x );
}}
```

% :

int // 4 Byte

:

~

```
class Demo {

public static void main(String args[]) {

    int i = -12;
    int j=13;

    i= i & j;

    i= i | j;
    i= (byte) j;


 System.out.println(i );
}}
```

-12 Binary

0...0 | 0...0 | 0...0 | 0000 1100 (12)
1...1 | 1...1 | 1...1 | 1111 0100 (-12)
0...0 | 0...0 | 0...0 | 0000 1101 (13)
0...0 | 0...0 | 0...0 | 0100 (4)

%i :

```
class Demo {

public static void main(String args[]) {

    int i = -12;
    int j=13;

    i= i & j;
    i= (byte) i;


 System.out.println(i );
}}
```

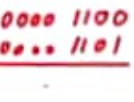
-12 Binary

0···0 | 0···0 | 0···0 | 0000 1100 (12)
1···1 | 1···1 | 1···1 | 1111 0100 (-12)

0

%i :

```
class Demo {

public static void main(String args[]) {

    int i = 12;
    int j=13;

    i= i& j;
    i= (byte) j;


 System.out.println(i );
}}
```



o/p : 12

CODING P ...

```
class Demo {

public static void main(String args[]) {

    int i = - 12;
    int j=13;

    i= i ^ j;

System.out.println(i ); //- 7
}}
```

0...0 | 0...0 | 0...0 | 0000 1100 (12)

1...1 | 1...1 | 1...1 | 1111 0100 (-12) } XOR →

0...0 | 0...0 | 0...0 | 0000 1101 (13) }

1...1 | 1...1 | 1...1 | 1111 1001 (-7)

=

-ve

2's complement

=

(-ve)

110

+ 1

---

111 (7)

CODING P ...

```
class Demo {

public static void main(String args[]) {

    int i = - 12;
    int j=13;

    i= i ^ j;

System.out.println(i ); //-7
}}
```

0...0 | 0...0 | 0...0 | 0000 1100 (12)

1...1 | 1...1 | 1...1 | 1111 0100 (-12)

0...0 | 0...0 | 0...0 | 0000 1101 (13)

XOR →

1...1 | 1...1 | 1...1 | 1111 1001 (-7)

=

-ve

2's c

```
class Demo {

public static void main(String args[]) {

    int i = 5;
    int j=i>>2;



 System.out.println(i );
}}
```

CODING P ...

```
class Demo {

public static void main(String args[]) {
    int i=9;
    Int k = 0;


    If((++i==1 ||++k==1))
       System.out.println(i); //10
    System.out.println(k);//
}}
```

CODING P •••

```
class Demo {

public static void main(String args[]) {
    int i=9;
    int k = 0;
    k= (i>k)? 10 : 20;
    k'= (i<k)? 10 : 20;

 System.out.println(k); //10
System.out.println(k'); //20
}}
```

CODING P •••

```
class Demo {

public static void main(String args[]) {

    int i = 5;
    int j=i>>2;


 System.out.println(j ); //1
}}
```

```
class Demo {

public static void main(String args[]) {

    int i = - 5;
    int j=i>>2;


 System.out.println(j ); //-2
}}
```

Exmp V.3

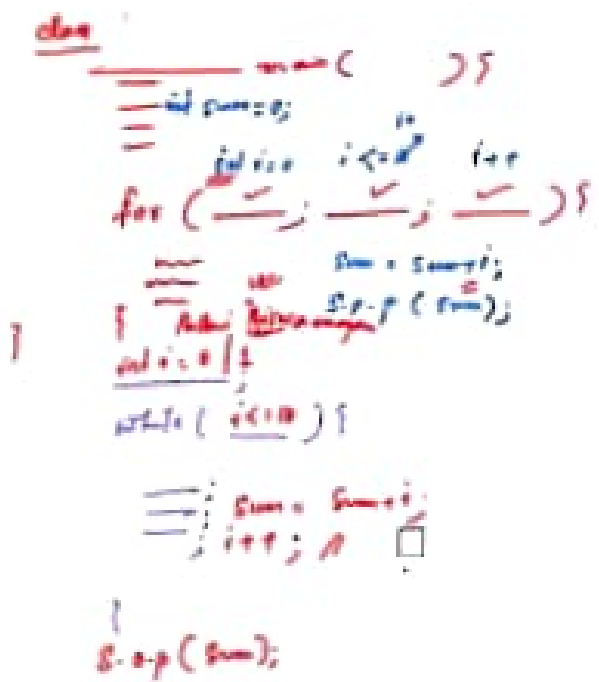
24 July 2023 18:03

```
public class Nestl {

    public static void main(String args[]) {

        int i=0, j=9;

        do {

            i++;

            if (j - - < i++) {
                break;
            } while (i < 5);

    System.out.println(i + "\t" + j);
    }
}
```

Output:

temp V.1

1. Reload the rest of the [illegible]



Output

0 1 2 | 0 1 | 1 2 1 2 1 2

```
public class Test {

    public static void main(String args[]) {

        int i=0, j=0;

        do {

            j++;

            if( i + j == i*i ) {
                break;
            }
        } while (j < 5);

    System.out.println(i + "" + j);
    }
}
```

% i = 6 j = 12

Exmp V.5

24 July 2021 18:30

```
public class Loop {

    static String o="";

    public static void main(String args[]) {
        Z:
        for (int x=2; x<7; x++) {
            if (x== 3) continue;

            if (x==5) break z;
            o=o+x;
        }
    System.out.println(o);
    }

}
```

Output:

2

4

Assign V.8

Q.8 Draw the following structure

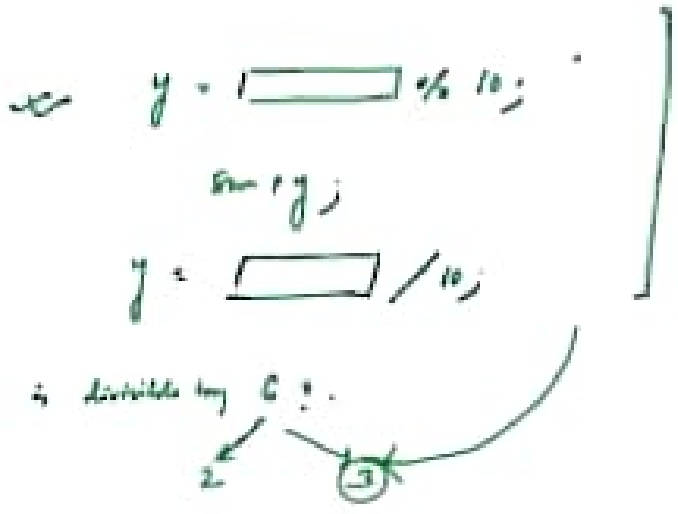
*
**
***
****
*****
******
****
****

2 Variable : i, j

for(i,j ; i< , i+

Assign V.)

4.) Last + number are last digit of that number →



11.1 Input a number and check whether it is an Armstrong number or not

Enter the • BufferReader class Syntax: in
• Scanner class
• Console

BufferReader (br) = new InputStreamReader (System.in));

(Syntax: in) // keyboard.

System.out.println("Enter the number");

```
public class [illegible] {
    public static void main(String[] args) {
        int n=3;
        int k=1;
        for (int i=0; i<3; i++) {
            for (int j = 0; j<n-i-1; j++) {
                System.out.print(" ");
            }
            for ( int z = 0; z < 2*i+1; z++, k++) {
                System.out.print(k);
            }
            System.out.println();
        }
    }
}
```

logic for space

logic for numbers

From <[illegible]>

1
2 3 4
5 6 7 8 9

i:0

j:0
j < n-i-1
< 3-0-1
< 2

i:1

i:2

z = 0
z < 2*i+1
0 < 2*0+1
0 < 1

< 2*i+1
<

Assign V.A

Q.8 Draw the following structure

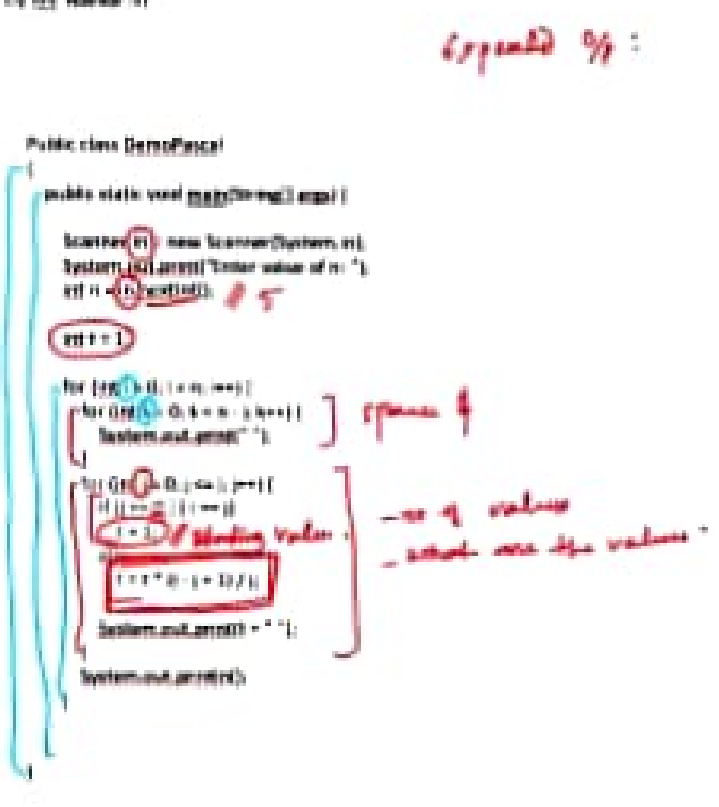
```
*
**
***
****
*****
******
*******
```

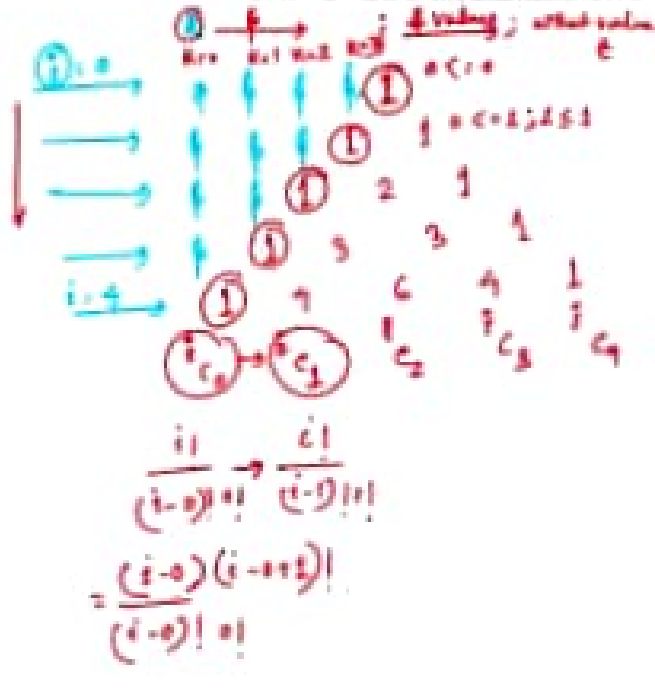
2 variable : i, j

```
for (i=0; i<7; i++){
    for(j=0; j<=i; j++){
        System.out.print("*");
    }
System.out.println();
}
```

```
}
public class Demo {
  }
  static int i;

    public static void main(String args[]){
        for(i=5;i>-5;i--)
            System.out.println(i);
        int i;
        for(i=1;i<=5;i++){
            System.out.print(i);
        System.out.println(Integer.i);
        }
    }
```

Pascal Pyramid by MK Rahul Sir



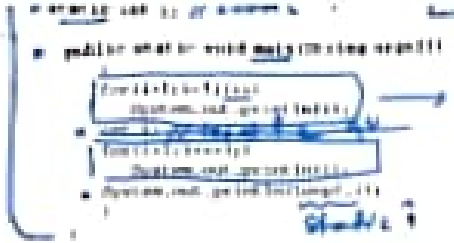


Pascal Pyramid by Mr Rahul Sir

```
Public class DemoPascal
{
  public static void main(String[] args){

    Scanner sc = new Scanner(System.in);
    System.out.print("Enter value of n: ");
    int n = sc.nextInt();

    int t = 1;

    for (int i = 0; i < n; i++){
      for (int k = 0; k < n - i; k++){
        System.out.print(" ");
      }
      for (int j = 0; j <= i; j++){
        if (j == 0 || i == j)
          t = 1;
        else
          t = t * (i - j + 1) / j;

        System.out.print(t + " ");
      }
      System.out.println();
    }

  }
}
```

Expected O/P :

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```



i : 1 ; i < 3
1 < 3 T Print (1)

i : 1++ 2 2 < 3 T Print (2)

i = 2++ = 3 3 < 3 F X

---

i = 1 ++i < 3
(1)++ < 3 T; Print (2)
2

i = 2 (2)++ < 3 T; Print (3)
3

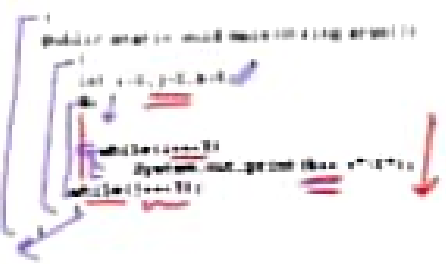
i : 3 3++ < 3 F X
4

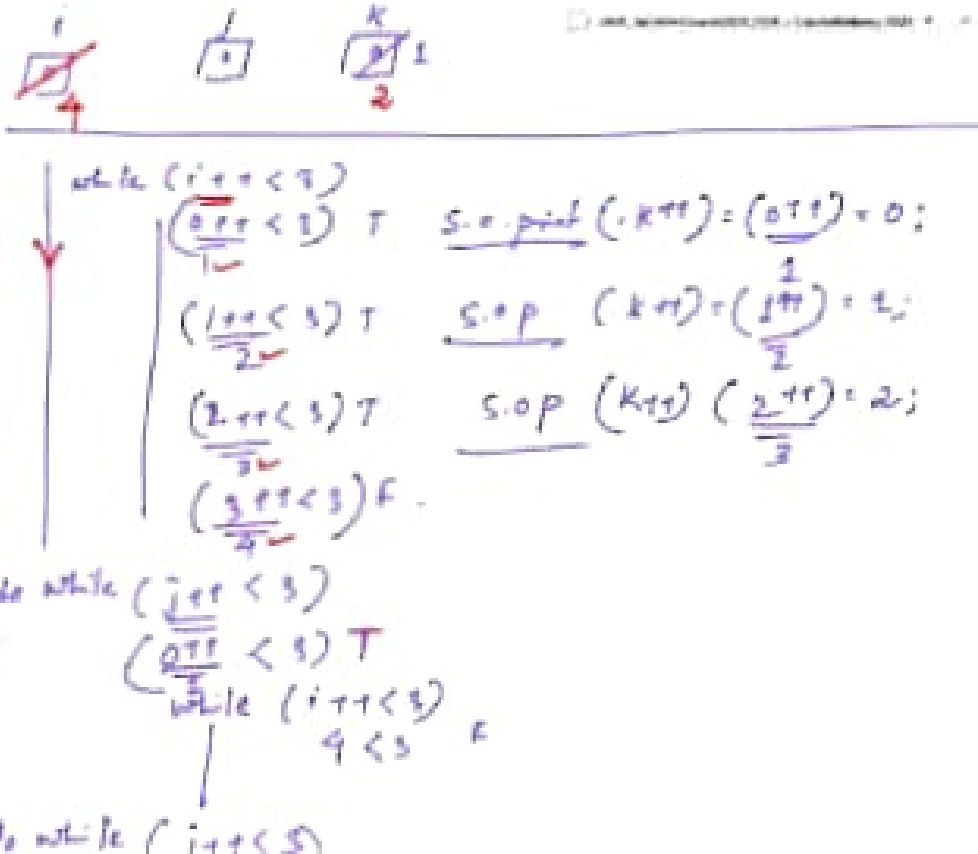
Sy s out println (Lmp1.i) ; // (3)

Exercise V.2

2.

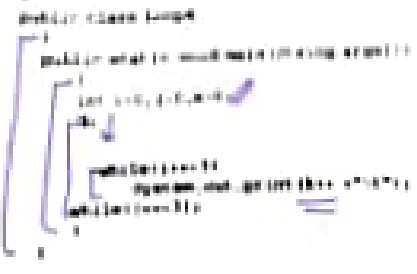
```
public class Loop2
   {
   public static void main(String args[])
        {
        int i=0;
        do
          {for(int i=0;i<=2;)
          System.out.print(i++*i);
        while(i++<=2);
        }
   }
```




i
j
k
while (i++ < 3)
(0++ < 2) T
S.o.println (k++) = (0++) = 0;
(1++ < 2) T
S.o.p (k++) = (1++) = 1;
(2++ < 2) T
S.o.p (k++) (2++) = 2;
(3++ < 2) F
do while (j++ < 3)
(0++ < 2) T
while (i++ < 3)
4 < 3 F
do while (j++ < 3)

0 0 0 | 0 1 2

Compiler V.S



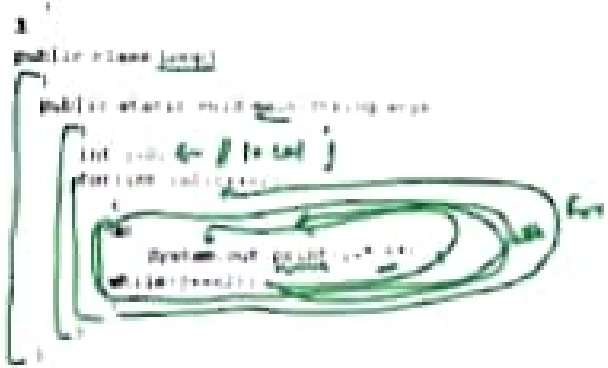
while (i++ < 3)

(0++ < 3) T S.o.println (i++) = (0++) = 0;

(1++ < 3) T S.o.p (i++) = (1++) = 1;

O/P

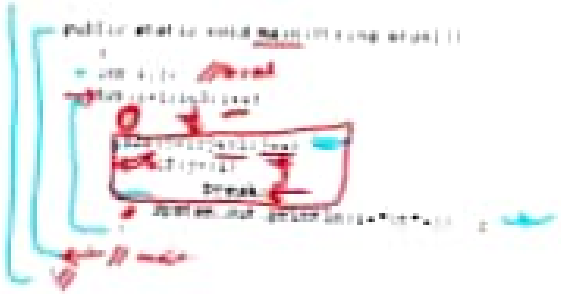
0 0 0 | 0 1 2 |



答: 1 1 1 2

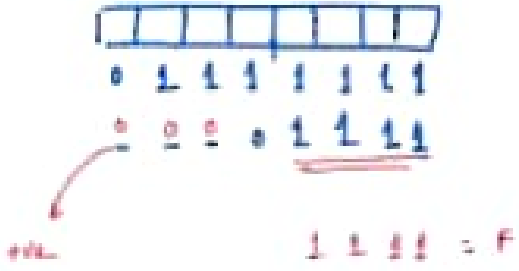
1
1
1
2

Print: 1
while (j++<2)
j++<2 T
1
Print: 1
while (j++<2)
j++<2 T
2
Print (i): 1
while 2++<2 F



Exercise IV.3

```
1. class Demo
   {
       public static void main(String [] args)
           {
               byte a=127;
               a++;
               System.out.println(a);
           }
   }
```

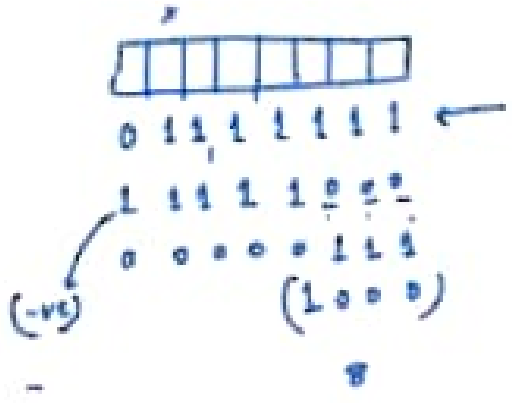


0/1

15

Exercise IV.1

```
[redacted]
{
    public static void main(String[] args)
    {
        byte a=127;
        a=a++;
        System.out.println(a);
    }
}
```



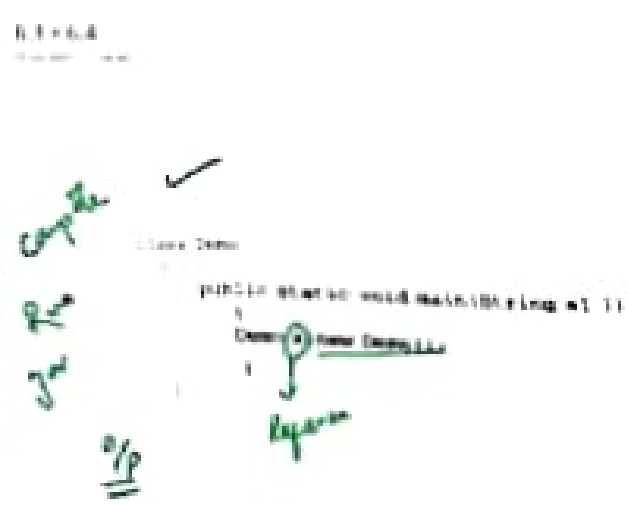
o/p

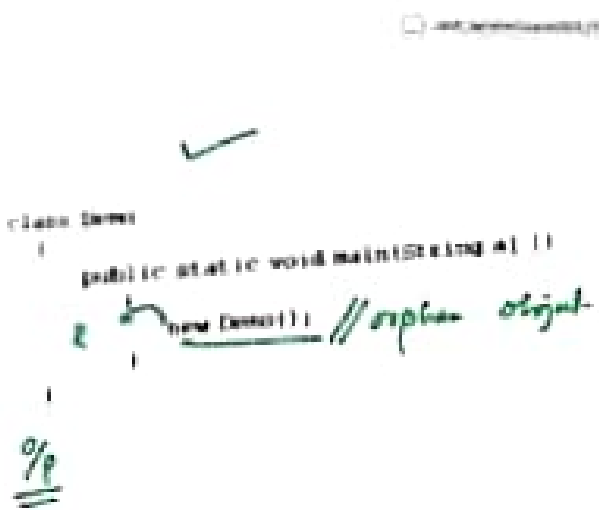
-8

Vs

```
8
public class Loop8
{
  public static void main(String args[])
    {
    a1:
    for(int i=0;i<3;i++)
      {
      a2:
      for(int j=0;j<3;j++)
        {
        if(i==j)
            break a1;
        System.out.println(i+" "+j);
        }
      }
    }
```

```
9
public class Loop9
{
  public static void main(String args[])
    {
    a1:
    for(int i=0;i<3;i++)
      {
      a2:
      for(int j=0;j<3;j++)
        {
        if(i==j)
            break;
        System.out.println(i+" "+j);
        }
      }
    }
```


public static void main(String a[])
new Demo();

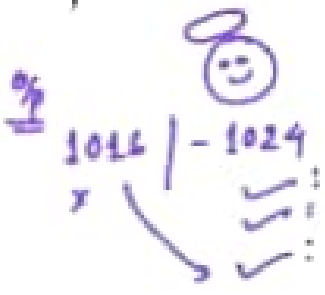

public static void main(String a[])
new Demo();
// orphan object

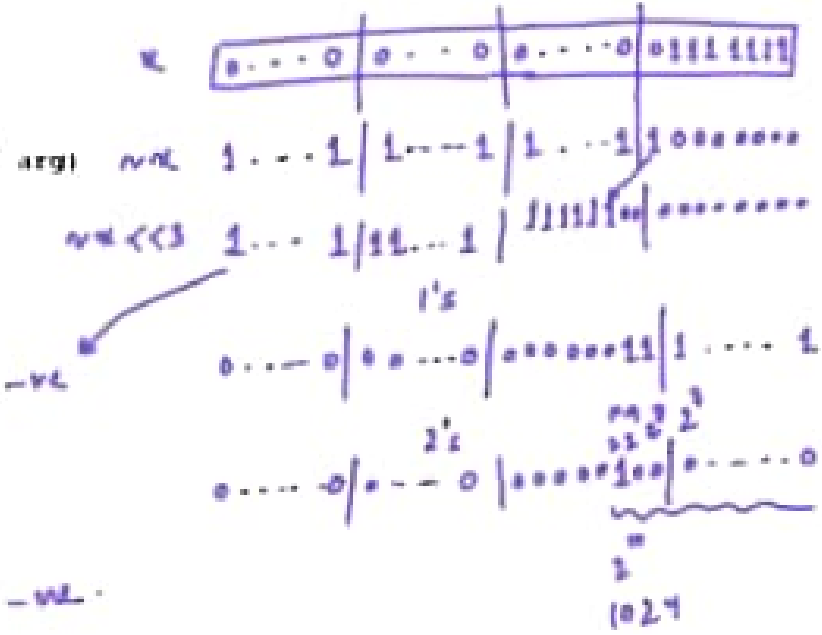
IV.6

a = -127

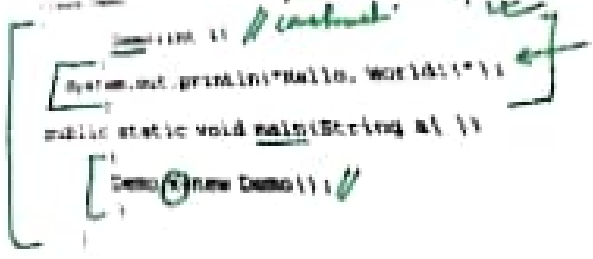
```
6. class Demo
   {
       public static void main(String [] args)
           {
               byte a=-127;
               a++;
               System.out.println(a);
           }
   }
```

```
class Demo
{
    public static void main(String [] args)
    {
        int x=127;
        x=x<<3;
        System.out.println(x);
    }
}
```





Day



O/P = error

```
class Demo {
}
public static void main(String arg[])
{
    Demo x = new Demo();
}
```

O/P No Error.

6.7

```
class Demo{
  public static void main(String[] ){
  }
     Demo a=new Demo();
     a.fun();
 }
 static void fun()
 {
    System.out.println("Hello");
    }
    void fun()
    {
       System.out.println("Hi");
       fun();
       Demo.fun();
       Demo a=new Demo ();
       a.fun();
     }
}
```

IV.9 TODO from this onwards

```
public class Demo
{
    public static void main(String [] args)
    {
        int a=1024;
        System.out.println(a);
    }
}
```

1024

```
public class Demo
{
    public static void main(String [] args)
    {
        int a=127;
        a=(a<<5);
        System.out.println(a);
    }
}
```

4192

```
public class Demo
{
    public static void main(String [] args)
    {
        int a=12;
        a=a<<2;
        System.out.println(a);
    }
}
```

48

```
public class Demo
{
    public static void main(String [] args)
    {
        int a=12;
        a=a<<3;
        System.out.println(a);
    }
}
```

96

IV.9 TODO from this onwards

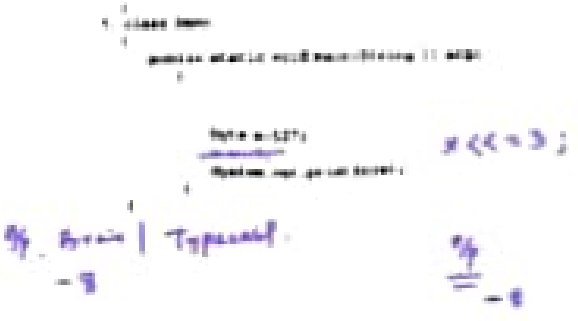
IV.12

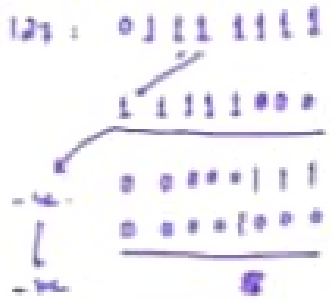
```
class Demo

public static void main(String[] args)
{
    int x=127;
    x=x>>>3;
    System.out.println(x);
}
```

```
x : (-1+x) >>> 3
  : (-1+127) >>> 3
  : 126 >>> 3
  : 2^7 >>> 3
  : 2^4
  : 16
```

Precedence	Operator	Type	Associativity
15	.	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ~ (type)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right
11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

IV.7





$x = 127 \quad y = 128$

$x = (x{+}{+} \mid 3)\ \&\ y$

$= (x{+}127 \mid 3)\ \&\ y$

1000 0011

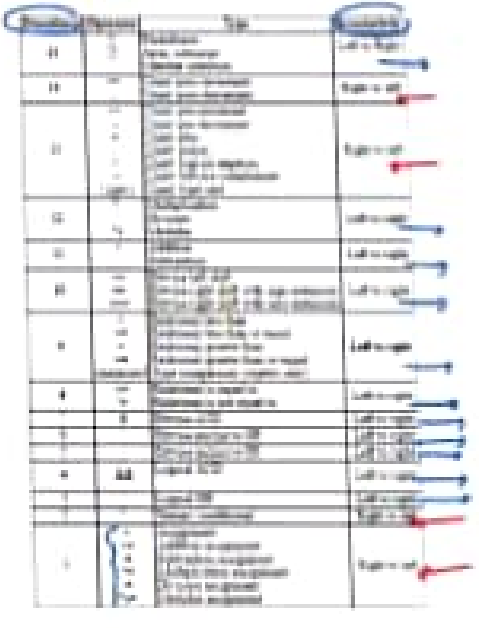
1000 0000

$= (128 \mid 3)\ \&\ 128$

$= 131\ \&\ 128$

$= 128$ Ans

Jack's



[illegible]) class Demo

```
{
    public static void main(String [] args)
    {
        int x=127, y=128;
        [illegible]
        System.out.println(x);
    }
}
```

ii) class Demo

```
{
    public static void main(String [] args)
    {
        int x=127, y=128;
        [illegible]
        System.out.println(x);
    }
}
```

[illegible]) class Demo

```
{
    public static void main(String [] args)
    {
        int x=127, y=128;
        [illegible]
        System.out.println(x);
    }
}
```

[illegible]) class Demo

```
{
    public static void main(String [] args)
    {
        int x=127, y=128;
        [illegible]
        System.out.println(x);
    }
}
```

= 128 << (131) >> 5   128

= [128 << (131 % 32)] >> 5   131

= (128 << 3) >> 5 $2^{10} \longrightarrow 2^{5}$

= (1024) >> 5

= (32) Ans BIG THANKS Rahul

Q of the Q. WORK   y = x++ << 3 + x++ >> 5

printf("x = %d %d", x, y); // x = 129   y = 32   ← HOME WORK

[y = x++ << 3 + x++ >> 5 ;]

128

= 128 << 3 + x++ >> 5;

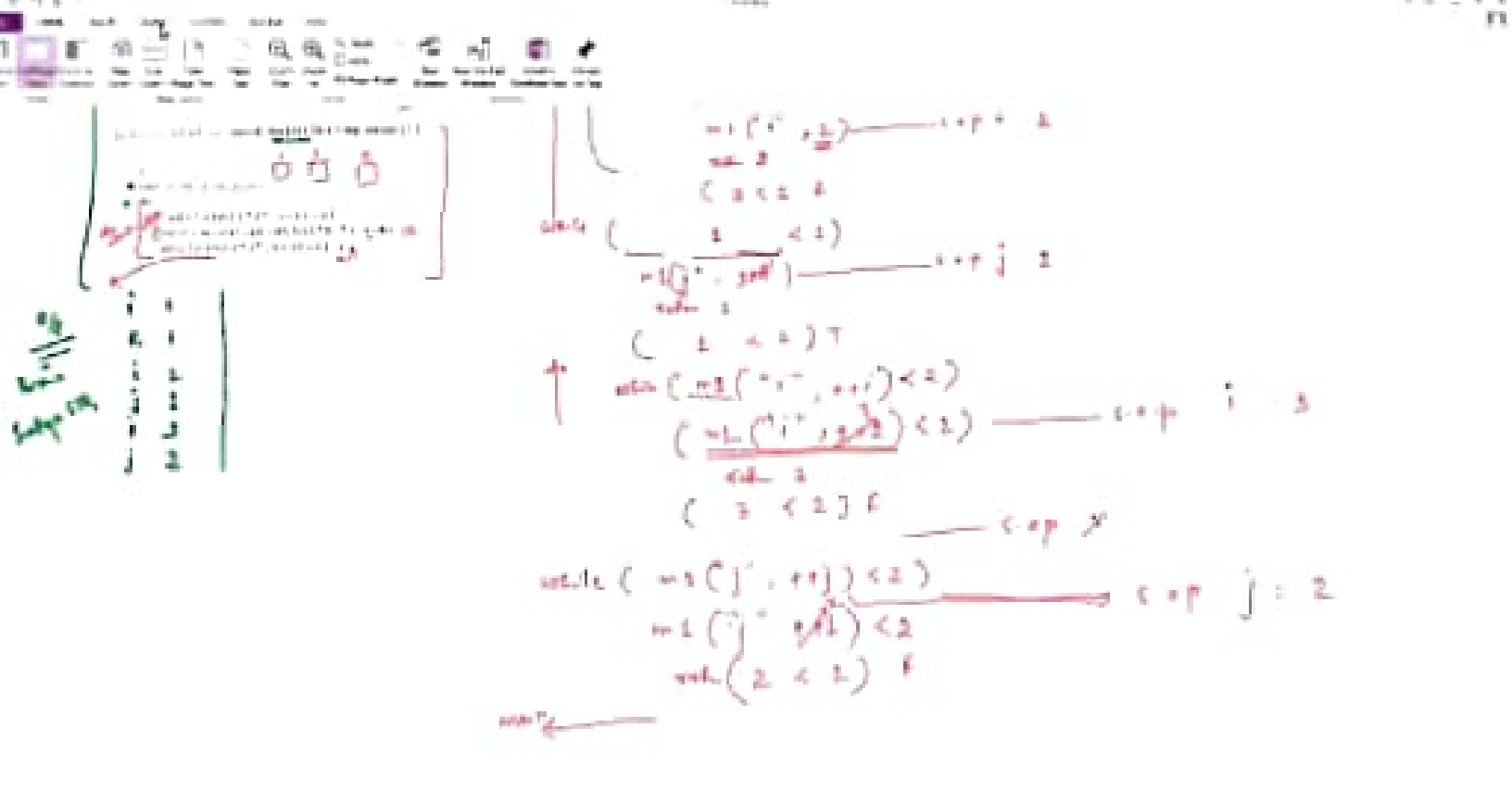
= 128 << (3 + 128) >> 5;

= 128 << (3 + 128) >> 5;

y

Exercise 11.4

```
(4)
public class Test4
   {
    void display()
        {
        System.out.println("Hello");
        }
    void display(int x)
        {
        System.out.println("display");
        }
    public static void main(String args[])
        {
        new Test4().display();
        }
   }
```




class Test4
{
void m1(String s) // method 1
{
System.out.println("Object");
}
void m1(String s) // method 2
{
System.out.println("String");
}
public static void main(String args[])
{
new Test4().m1(null);
}
}
m1(null)
o/p
String

A. Design a class such that only **one object** of that class can get created by any user.

B. Create an object of a class, which is having **only one private constructor** and who's **No other constructor can be defined.**

TRY?

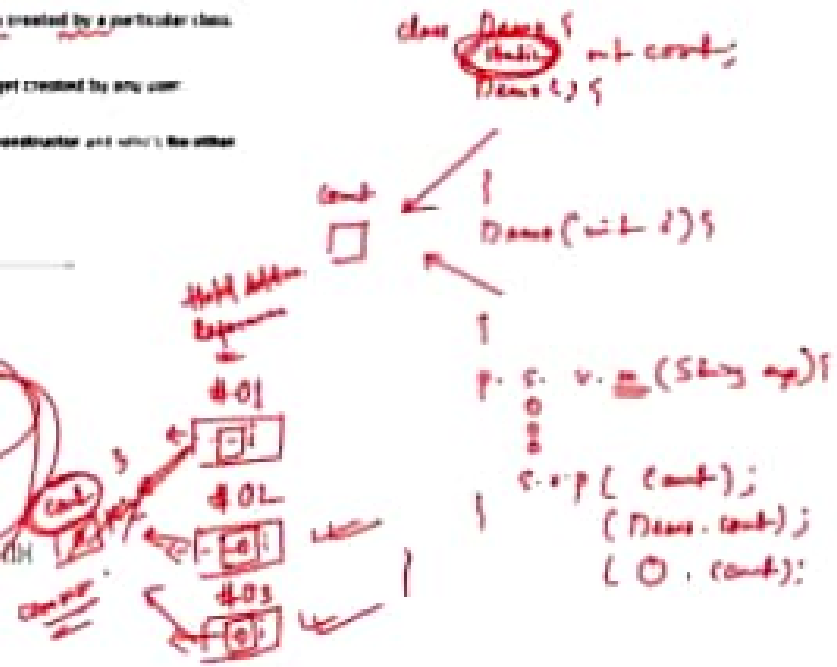
```
Class Demo {

    <static> boolean IsCreated; //0

    Demo(){
        Cancel();
    }

  static Demo  cancel( ){
        if (!Iscreated()
            return this
        else
            return null;
    }

    public static void main( ){
        Demo d1 = new Demo();
        Demo d2 = new Demo();
}
```

Q8. Write a JAVA program to Calculate the number of objects created by a particular class.

Q. Design a class such that only one object of that class can get created by any user.

Q. Create an object of a class, which is having only one private constructor and which the other constructor can be defined.

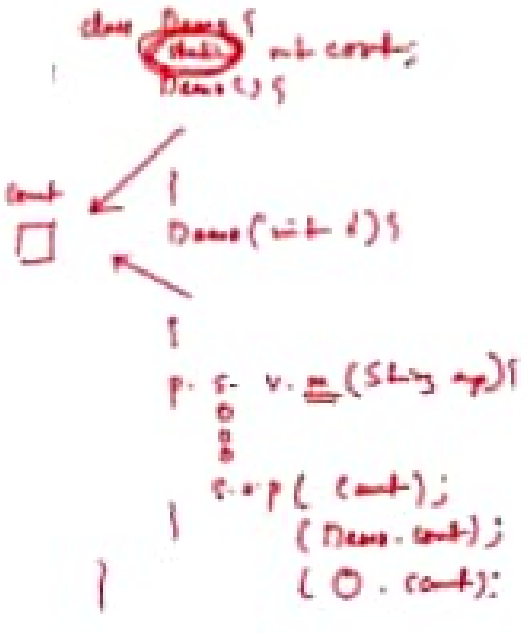
```
Class Demo {
    int i;
    Static int count = 0;// 0
    Demo(){
    Count = count + 1;
    }
    Demo(int a){

    }
    Public static void main(String args[]){
        Demo o1 = new Demo();
        Demo o2 = new Demo();
        Demo o3 = new Demo();
        Demo o4 = new Demo(10);
    }
```



Q8. Write a JAVA program to Calculate the number of objects created by a particular class

Q. Design a class such that only one object of that class can get created by any user.

Q. Create an object of a class, which is having only one private constructor and which's No other constructor can be defined



```
//@
public class Test7
{
  int i;
  Test7()
  {
    System.out.println("Java");
  }
  Test7(int i)
  {
    this.i=i;
    this();
    System.out.println(i);
  }
  public static void main(String args[])
  {
    new Test7(7);
  }
}
```

O/p
Java
7

this() // Empty argument based Const.

4. * Design a class such that only one object of that class can get created by any user.

(5) * Create an object of a class, which is having only one private constructor and which No other constructor can be defined.

TRY?

```
class Demo {

    static boolean isCreated; //0

    Demo(){
        cancel();
    }

    static Demo cancel(){
        if(!isCreated)
            return this;
        else return null;
    }

    public static void main(){
        Demo o1 = new Demo();
        Demo o2 = new Demo();
    }
```

Q. A. * Design a class such that only one object of that class can get created by any user.

* Create an object of a class, which is having only one private constructor and which the other construction can be defined.

TRY?

```
class Demo {

    Demo() {
        cancel();
    }

    static Demo cancel() {
        if (T.counter)
            return this;
        else return null;
    }

    public static void main() {
        Demo o1 = new Demo();
        Demo o2 = new Demo();
    }
}
```

```
public class CastI{
    public static void main(String args[]){

    int i=10;

    long l=1;

    double d=3.4;

    byte b;
    b= (byte)i;
    System.out.println(b);

    i= (int) l;
    System.out.println(i);

    l=(long) d;
    System.out.println(l);
```

File: xyz.java

```
class Hello {
    public static void main (String args []) {
        System.out.println("Hello");
    }
}
```

File: hello.java

```
public class hello {
    public static void main (String ag []) {
        System.out.println("Hello");
    }
}
```

Rough

Rebuild { Compile : javac XYZ.java
Run : java hello.

{ Compile : javac hello.ja
{ Run : java hello.

```
public class CastI{
    public static void main(String args[])

    int i=10;

    long 1=1;

    double d=3.4;

    byte b;
    b= (byte)i;
    System.out.println(b);

    j= (int) 1;
    System.out.println(i);

    1=(long) d;
```

(Ctrl) ▾

```
class Demo {

public static void main(String args[]) {

int x=12;

System.out.println(~x);

}}
```

o/p -12 | -13

```
public class Cast{
    public static void main(String args[]){

        int i=10;

        long l=1;

        double d=3.4;

        byte b;
        b= (byte)i;
        System.out.println(b);

        i= (int) l;
        System.out.println(i);

        l=(long) d;
        System.out.println(l);
    }

}
```

b = 10

// i = 1

// d = 3

```
class Demo {

public static void main(String args[]) {

    int i = - 12;
    int j=13;

    j= i ^ j;


 System.out.println(i );
}}
```

```
class Demo {

public static void main(String args[]) {

    int i = - 12;
    int j=13;

    i= i & j;
    i= (byte) i;


 System.out.println(i );
}}
```

-12 Binary →

0000 1100 (12)
110. (-12

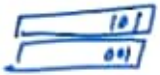
%i :

```
class Demo {

public static void main(String args[]) {

      byte i=10;
      byte j=9;
      int i=j+i;
      i= (byte) (j+i);
      i= (byte) j+ (byte)i;

  System.out.println(i );
}}
```

```
class Demo {

public static void main(String args[]) {
    int i=64;
    int k = 0;
    int k'=0;
    k= (i>k)? 10 : ---;
    k= (i>k)? --- : ---;

  System.out.println(k); //10
  System.out.println(k'); //10
}}
```

CODING P ...

```
class Demo {

public static void main(String args[]) {

    int i = 5;
    int j=i>>2;

 System.out.println(j ); //1
}}
```




```
class Demo2

public static void main(String args[]) {

    int i = - 5;
    int j=i>>2;

 System.out.println(j ); //-2
}}
```

LSB

11 . . . 1 | 1 . . . . 1 | 1 . . . 1 | 11111011

11111110

1 . . . . 1 |

↓

-ve

2's complement

-ve

001

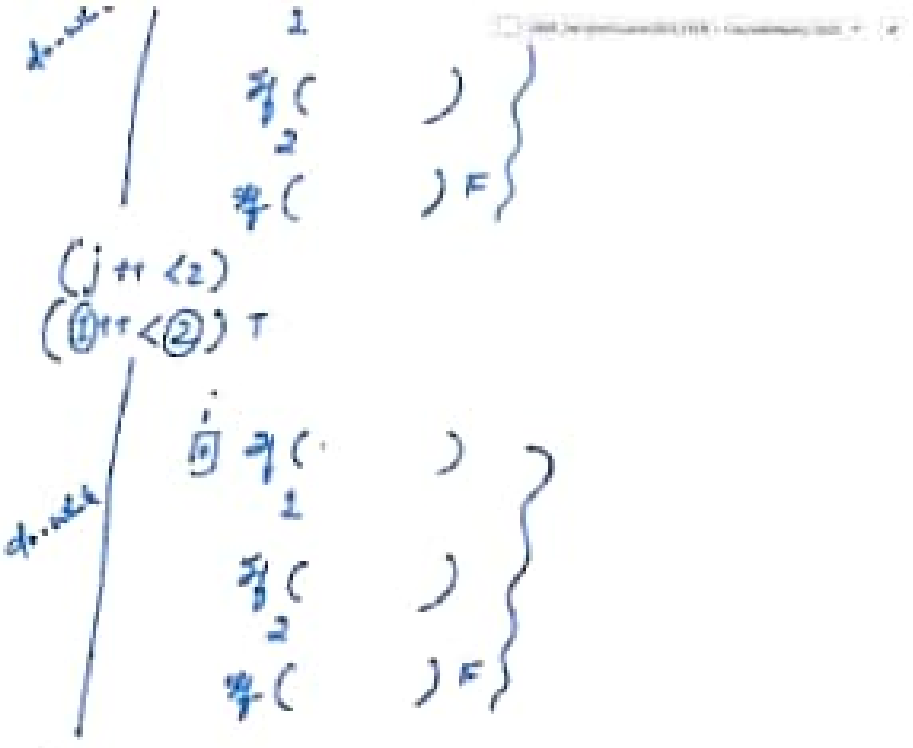
+1

010

```
class Test2 {
    public static void main(String args[]) {
        int s=0;
        for (int n=1;n<=10; n++) {
            s=s+n;
        }
    System.out.println("Summation of the first ten natural no. : "+s);
}
}
```

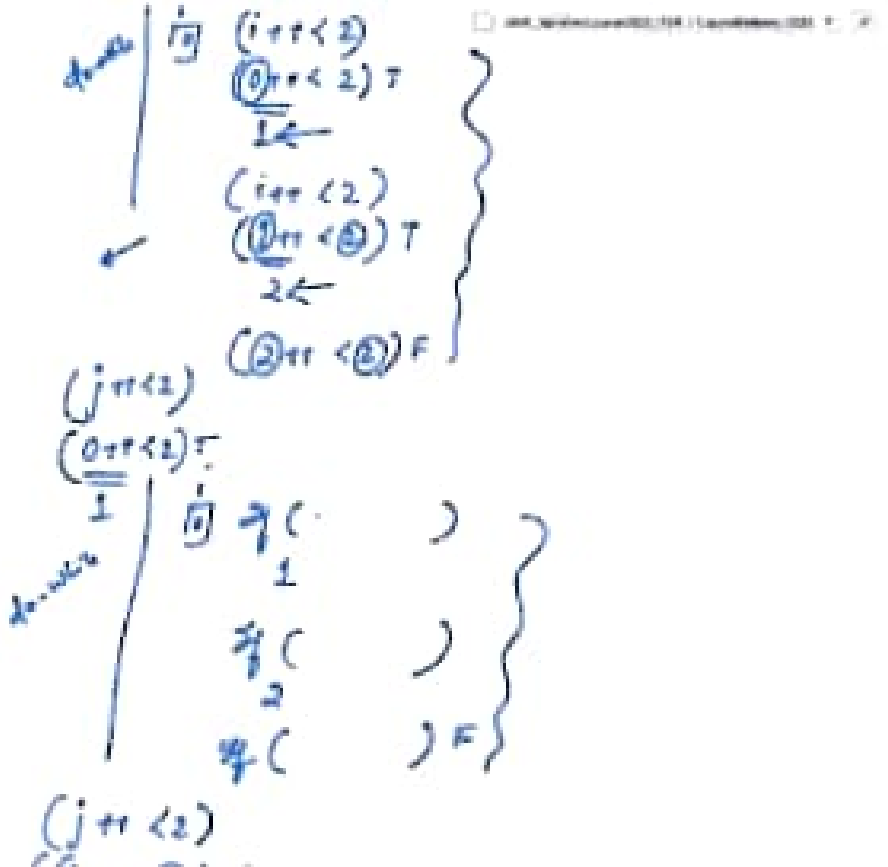
```
class Demo {

public static void main(String args[]) {
    int i=9;
    Int k = 0;


    If((++i==1 ||++k==1))

 System.out.println(ϕ: //10
}}
```

CODING P •••




public static void main(String args[]) {
}


Output



Q. Explain the use of the ... using ...

```
int i = 1; // initialization
do {
    ____ ; sum = sum + i;
    ____ ; i++; // increment
} while ( i <= n );
Sc.p(sum);
```

```
int sum = 0;
for ( int i = 0 ; i <= n ; i++ )
{
    sum = sum + i;
    Sc.p(sum);
}
```

```
int i = 1;
while ( i <= n )
{
    sum = sum + i;
    i++; // increment
}
Sc.p(sum);
```

Q.5 Enter a number and check whether it is a [illegible] number or not.

s.nextInt();

Assign V.4

Q4 Enter a character and print the ASCII value of that number

Ex:

```
Scanner s = new Scanner(System.in);
char c = s.next().charAt(0);  // Thank you
- int i = (int) c;
byte
```

Assign V.S

Q.9

Q1 Show the following structure

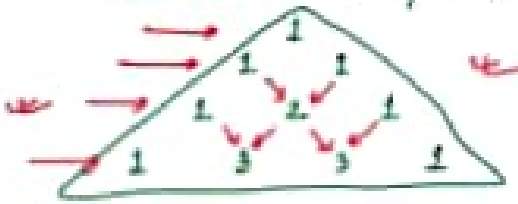
n

```
for (i=0; i<n ; i++)
    for (j=0; j<n-1 ; j++)
        {
          cout("  ");
        }
    [ for (k=0; k<=i; k++){
      [
      }
```

group

Q.10

Question : Pascal Pyramid/Triangle



```
for (i=0; i<n; i++ {
    for (j=0; j<=i; j++){
      cout (" ");
    }
    for (k=0; k<=i; k++){
      cout (
    }
```


Think Java

Pascal Pyramid by Ajit Rathod

Expected O/P :

```
Public class DemoPascal
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter value of n: ");
    int n = sc.nextInt();  // 5

    int t = 1;

    for (int i = 0; i < n; i++) {
      for (int k = 0; k < n - i; k++) {
        System.out.print(" ");
      }
      for (int j = 0; j <= i; j++) {
        if (j == 0 || i == j)
          t = 1;
        else
          t = t * (i - j + 1) / j;

        System.out.print(t + " ");
      }
      System.out.println();
    }

  }
}
```

Q 9 Show the following structure

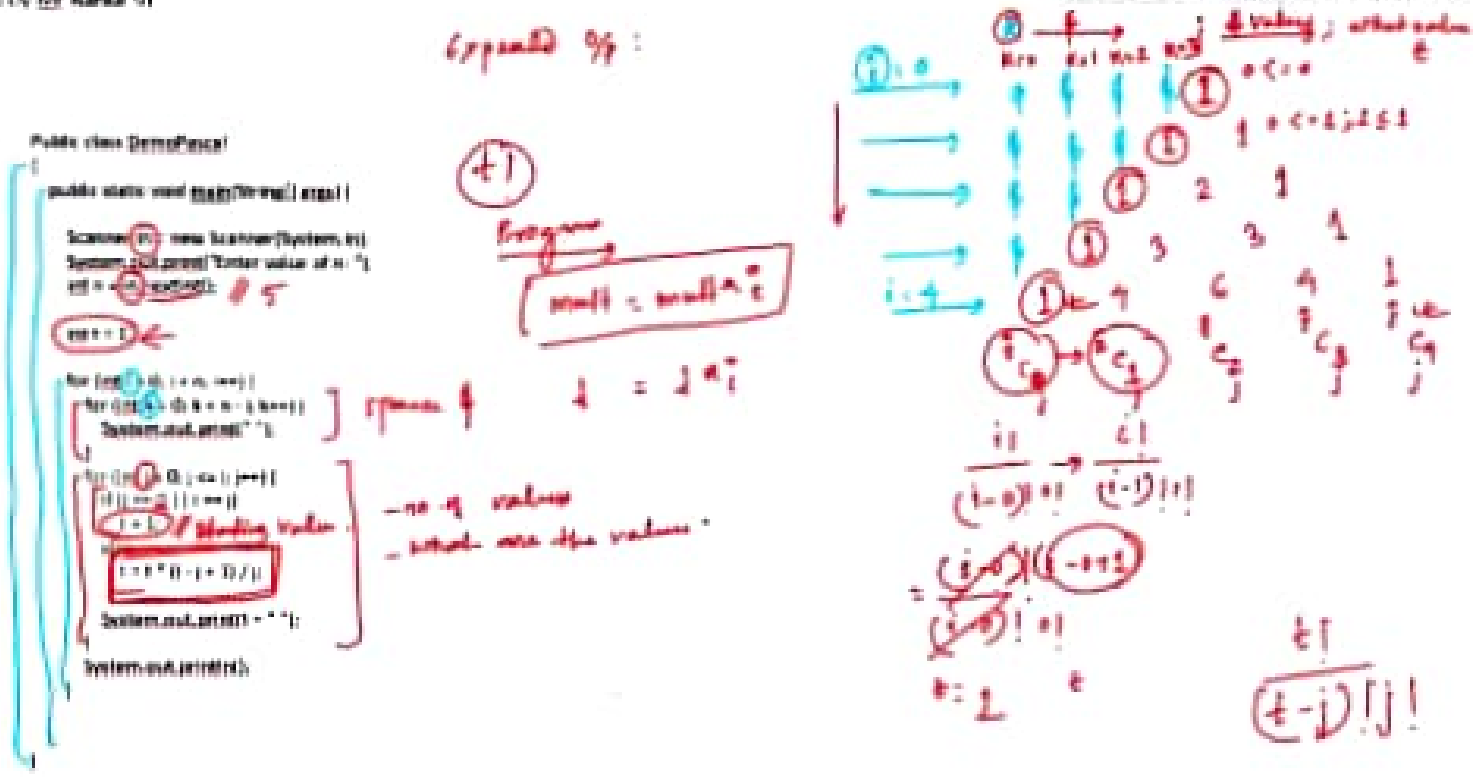
```
        1
    2   3   4
5   6   7   8   9
```

By Rahul Sir

```
public class Loop {
public static void main(String[] args) {
int n=3;
int k=1;
for (int i=0; i <3; i++) {
for (int j = 0; j<n-i-1; j++) {
System.out.print(" ");
}

for ( int j = 0; j < 2*i +1; j++,k++) {
System.out.print(k+" ");
}
System.out.println();
}
}
}
```

From <[illegible]>

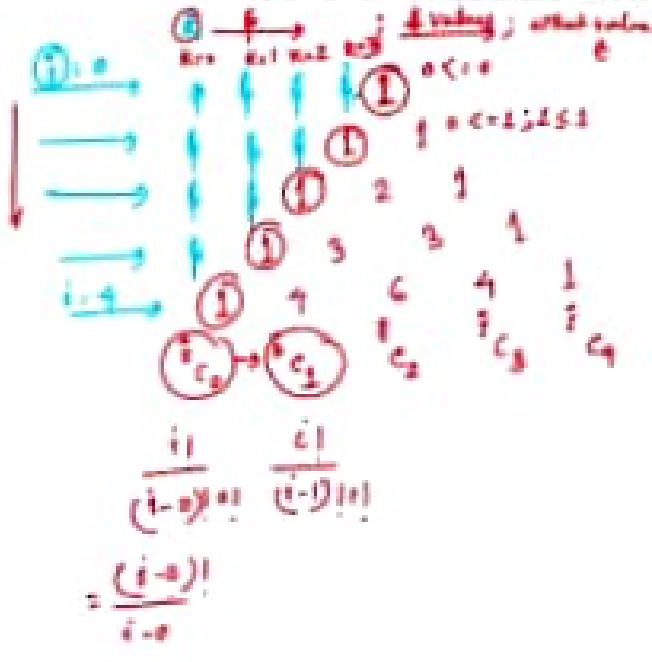
Pascal Pyramid by Mr. Rahul Vi



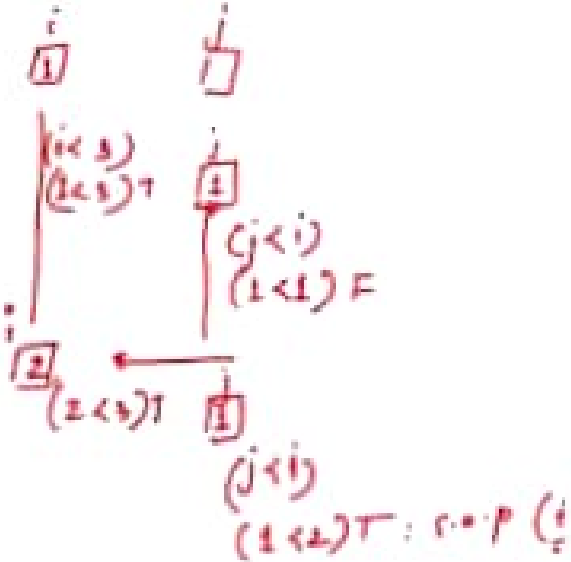
Pascal Pyramid by Mr. Rahul Sir

Explanation %:

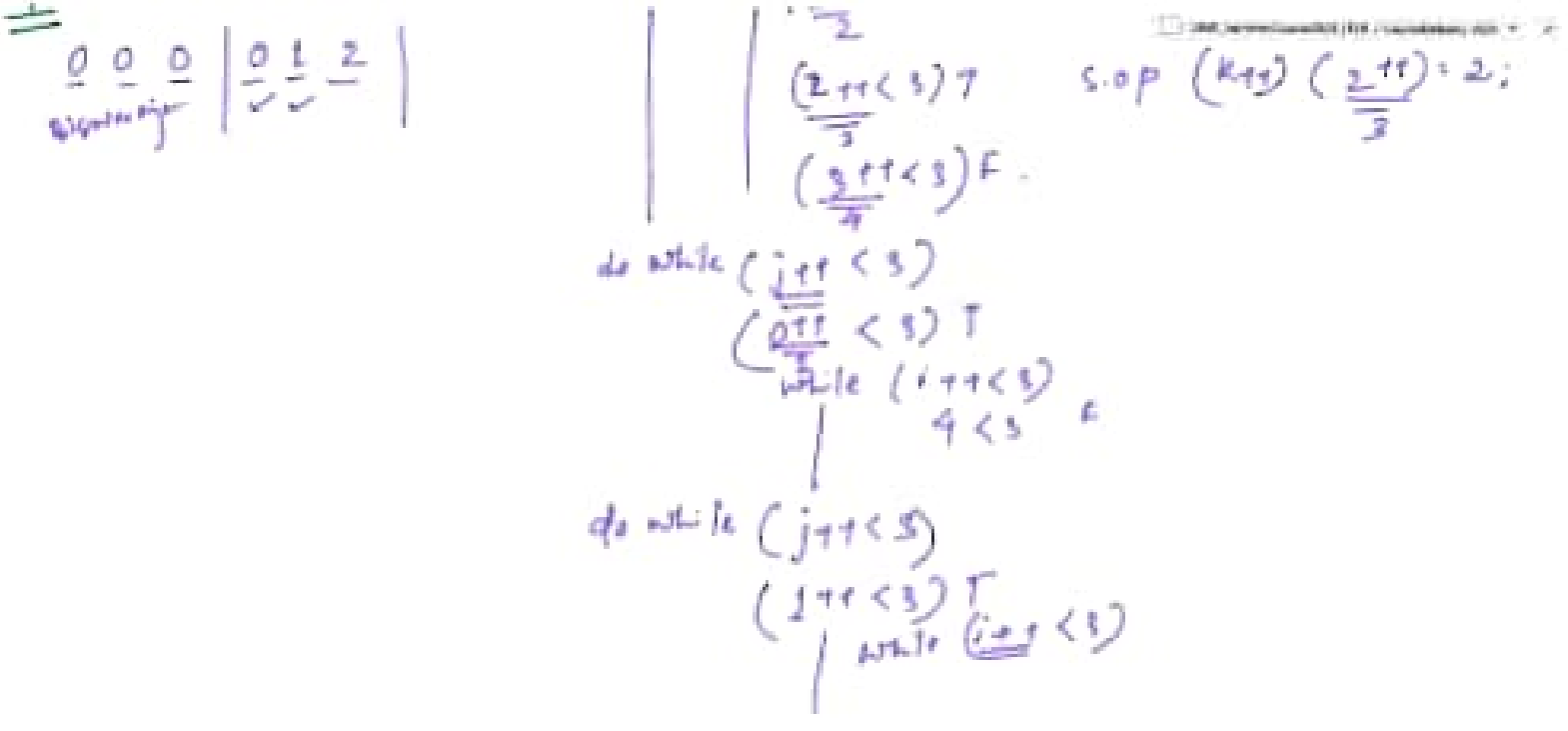
```
Public class DemoPascal
{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value of n: ");
        int n = sc.nextInt();
        int t = 1;
        for (int i = 0; i < n; i++) {
            for (int k = n - i; k >= 0; k--) {
                System.out.print(" ");
            }
            for (int j = 0; j <= i; j++) {
                if (i == j || i == 0) {
                    t = 1;
                } else {
                    t = t * (i - j + 1) / j;
                }
                System.out.print(t + " ");
            }
            System.out.println();
        }
    }
}
```



```
public class Loops
{
    public static void main(String a[])
    {
        int i,j; // uninitialized Local i & local j
        for(i=1;i<=3;i++) // Outer for i val
        {
            for(j=1;j<i;j++) // Inner for i val
            {
                System.out.print(i+" "+j+" "); //
            }
        }
    }
}
```

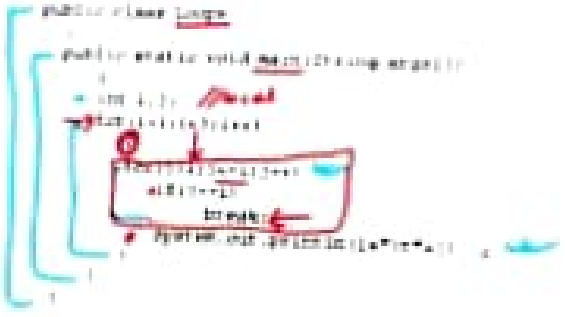


Op: 2 1


0 0 0
0 1 2
Beginning
2
(2++<3) T
(3++<3) F.
S.o.p (k++) (2++) = 2;
do while (j++ < 3)
(0++ < 3) T
while (i++<3)
4 < 3 F
do while (j++<3)
(j++<3) T
while (i++ <3)

9

```
9
public class Loop9
   {
   public static void main(String args[])
       {
       a1:
       for(int i=0;i<3;i++)
          {
          a2:
          for(int j=0;j<3;j++)
              {
              if(i==j)
                    continue a1;

              System.out.println(i+"\t"+j);
              }
          }
       }
   }
```



2 1 | 1 1 | 1 1
2 1 | 2 2 | 2 2

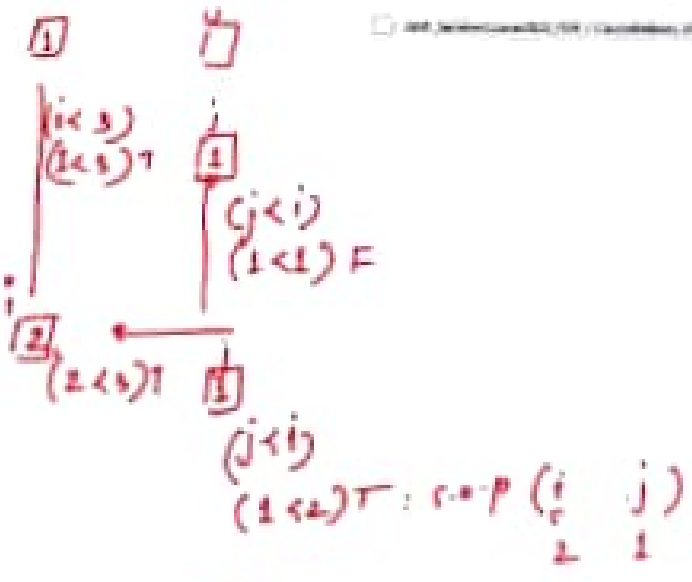
(i<3) T; j=1
(j<=i) (1<=1) T
print (j==i) 1==1

S.o.p (i, j) // 1 1

(i<3) 2<3 T : j=1
(j<=i) (1<=2) T print (j==i) (1==2)x

```
public class Tough
{
    public static void main(String ar[])
    {
        int i,j; // uninitialized  Local i & local j
        for(i=1;i<3;i++) // Outer-for = 2 times
        {
            for(j=1;j<i;j++) // Inner-for = 1 time
            {
                System.out.print(i+":"+j+" ");
            }
        }
    }
}
```

Op: 2 1



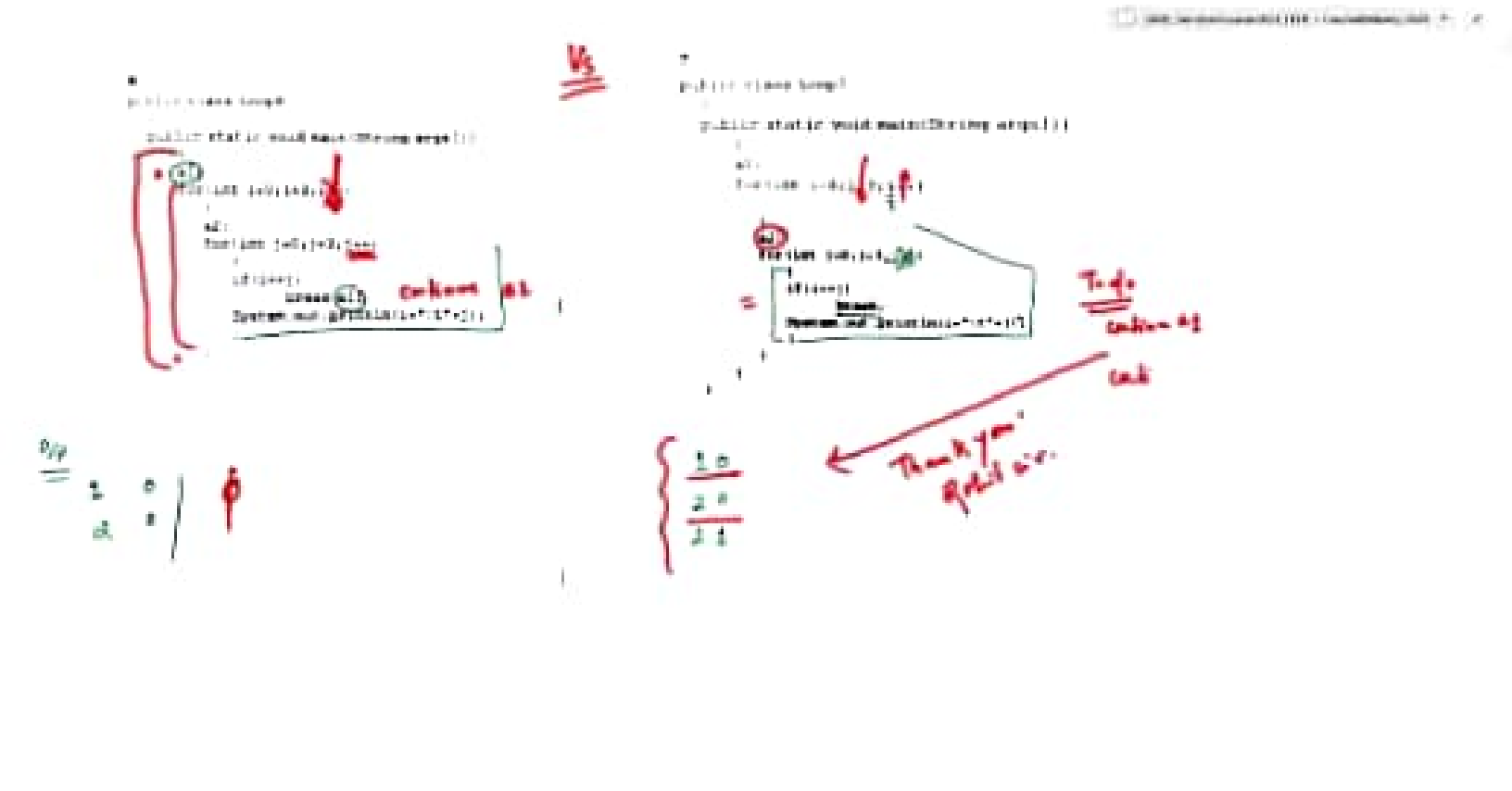
Exercise IV.1

```
[redacted] class [redacted]
{
    public static void main(String [] args)
    {
        byte a=127;
        a=a++;
        System.out.println(a);
    }
}
```

o/p

255
- 8

248



2

```
class Demo
{
  public static void main(String args[ ])
      {
      int x=3;
        {
        int x=7;
        System.out.println(x);
        }
      System.out.println(x);
      }
}
```

O/P

Error : Duplicate local variable

6.2

```
class Demo
{
  public static void main(String args[ ])
      {
      int a=1;
         {

          int a=7;
          System.out.println(a);
             }
          System.out.println(a);
          }
```

IV.7

```
1. class Demo
   {
      public static void main(String [] args)
         {

            byte a=127;
            System.out.println(a);
         }
   }
```

x <<= 3 ;

% . Arrow | Typecast.

-8

%

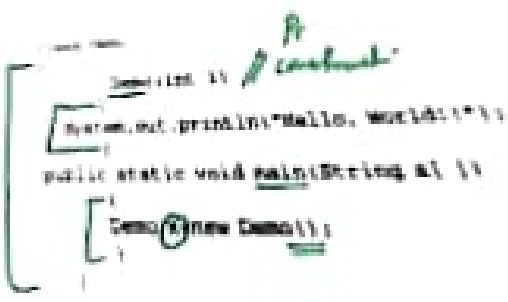
6.7

27 July 2023 09:53

```
class Demo{
  public static void main(String f[ ])
  {
     Demo x=new Demo();
     x.fun1();
  }
 static void fun()
 {
    System.out.println("Hello!");
    }
    void fun1()
    {
       System.out.println("Hi");
       fun();
       Demo.fun();
       Demo x=new Demo ();
       x.fun();
    }
 }
```

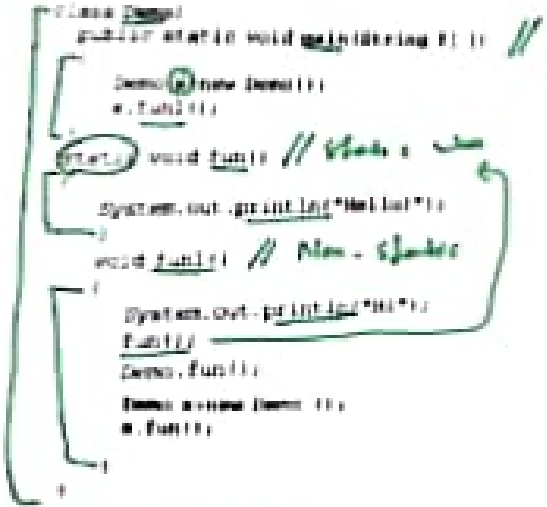
o/p

Hi
Hello
Hello
Hello .


// constructor
System.out.println("Hello, World!");
public static void main(String a[]
new Demo();

O/p :- error

Precedence	Operator	Type	Associativity
15	() [] .	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ! ~ (type)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right
11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	?:	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left



o/p

Hi
Hello
Hello
Hello

N.1)

IV.13

Associativity: { << , >> }

++ Right to Left

<< | >> Left Right

+ Right to Left

x = (++a << 3 + x++) >> 5 ;

= ++a (123) << (3 + (123++)) >> 5 ;

= (124) << (3 + (123)) >> 5

= 124 << (131) >> 5

= [124 << (131 % 32)] >> 5

= (124 << 3) >> 5

= (1024) >> 5

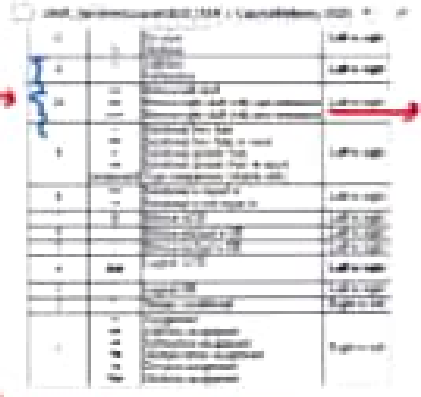
= 32 Ans

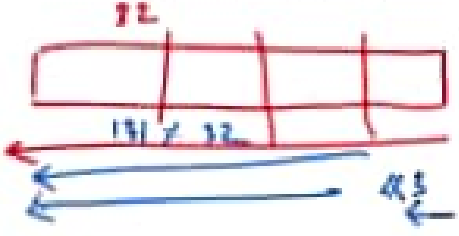
124

123

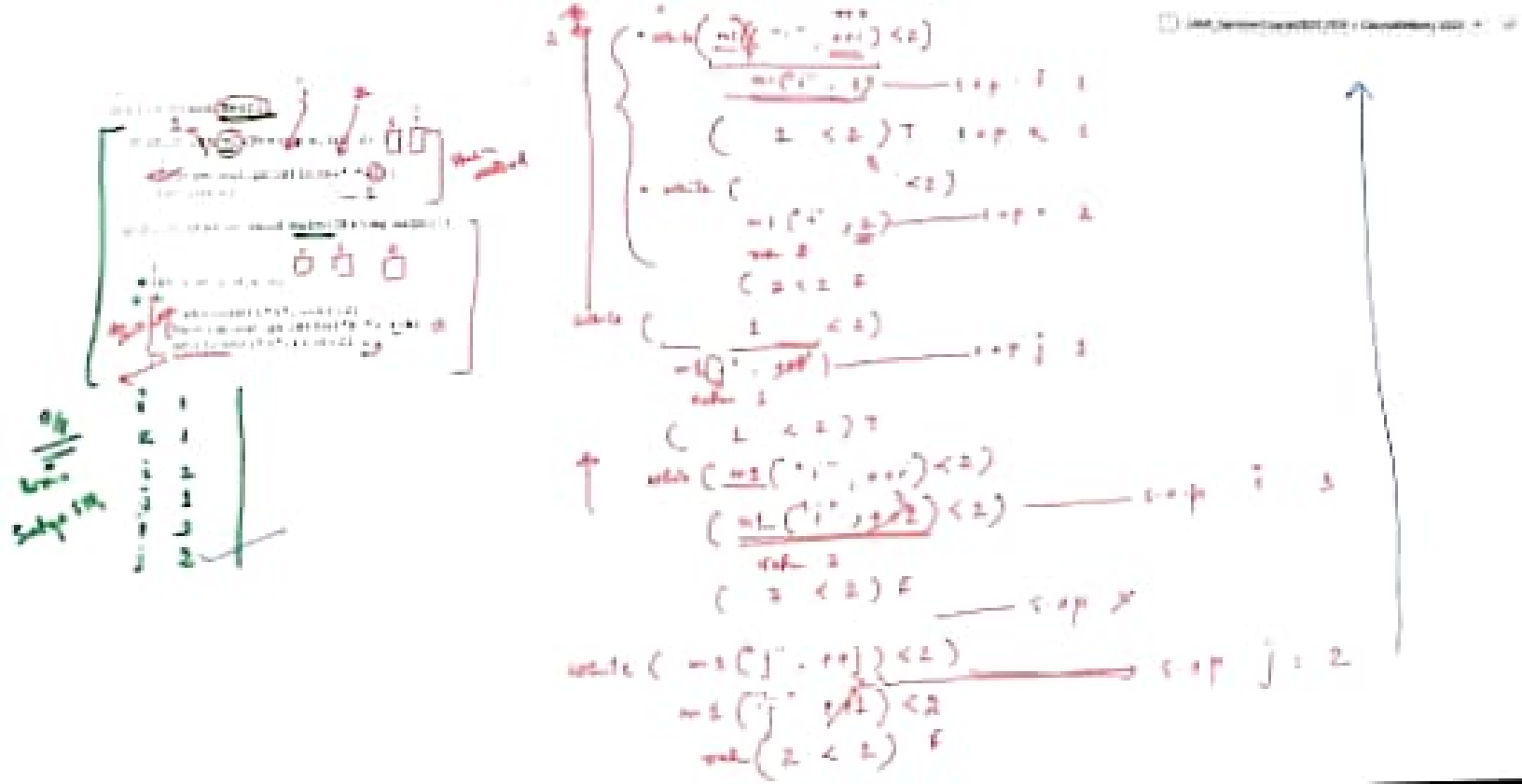
131

$2^{10} \longrightarrow 2^5$

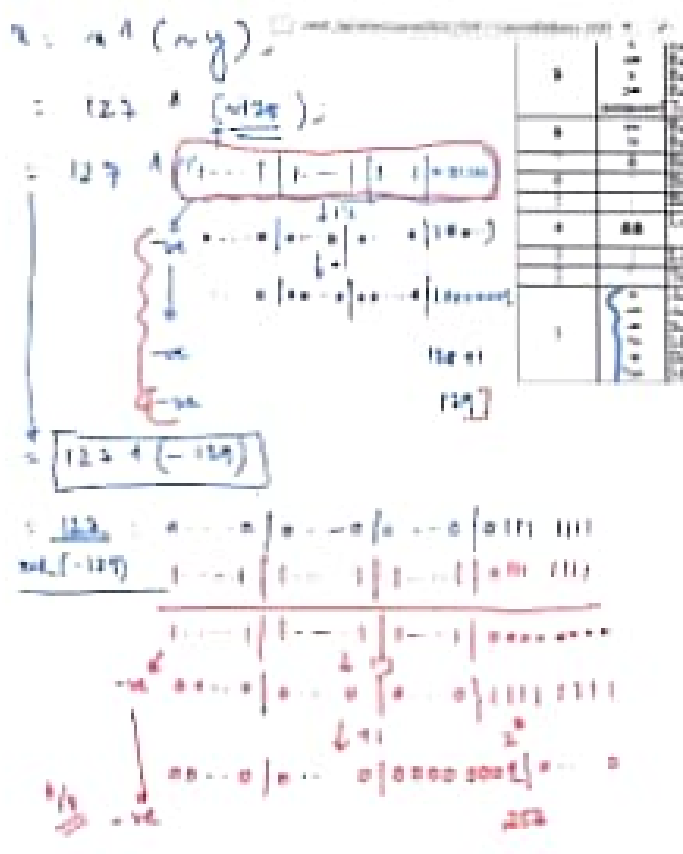


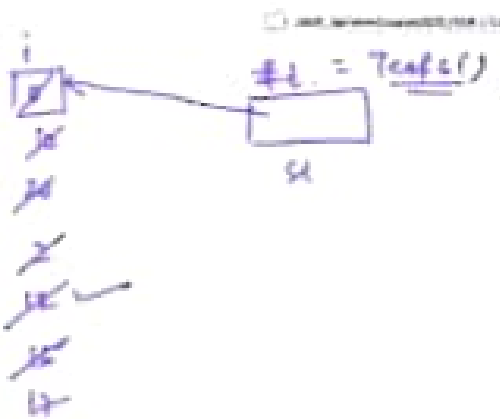


BIG THANKS Rahul



```
        System.out.println(a);
    }
}

class Test {

    public static void main(String[] args)
    {
        int x=127, y=129;
        x=(int)(byte)x;
        System.out.println(x);
    }
}

class Demo
{
    public static void main(String[] args)
    {
        int x=127, y=129;
        x=(byte)x;
        System.out.println(x);
    }
}
```





```
{!
public class Test6
    {
    static int i;
    public Test6()
        {
        i-=4;
        System.out.println(i);
        }
    void show()
        {
        i++;
        System.out.println(i);
        }
    {
        i+=10;
        System.out.println(i);
    }
    static{
        i+=10;
        System.out.println(i);
                }
    public static void main(String args[])
        {
        i/=10;
        System.out.println(i);
        Test6 t=new Test6();
        t.show();
        }
     static{
       i+=10;
       System.out.println(i);
             }
    }
```

```
}
```

Q. Design a class such that only **one object** of that class can get created by any user

Q. Create an object of a class, which is having **only one private constructor** and other's **No other constructor can be defined**

```
Class Demo {
    int i;
    Static int count = 0;// 0
    Demo(){
        count = count +1;
    }
    Demo(int x){
        i = x;
        count = count +1;
    }
    Public static void main(String args []){
        Demo o1 = new Demo();
        Demo o2 = new Demo(x
        Demo o3 = new Demo(y
        Demo o4 = new Demo(10);
    }
}
```