Output Questions

1.	public class Test {
	public static void main(String[] args) {
	System.*out*.println(System.*out*.println("hi"));
	}
	}

Answer:	*void type error*
Explation:	The return type of System.*out*.println is void type means it return nothing

2.	public class Test {
	public static void main(String[] args) {
	int a, b, c;
	a=-3+2*7-4;
	b=a*8+4%5-6;
	c=a+b*3-2%5-4;
	System.*out*.println(a+" "+b+" "+c);
	}
	}

Answer:	7 54 163
Explanation:	Simple Arithmetic operations

3.	public class Test {
	public static void main(String[] args) {
	int a=2,b=5,c;
	a=a*a++ - --a;
	c=b++ - b--;
	System.*out*.println("a="+a+",b="+b+",c="+c);
	System.*out*.println(a++ + ++a * a--);
	System.*out*.println(b=b++ * b--);
	System.*out*.println("a="+a+",b="+b+",c="+c);
	}
	}

Answer:	a=2, b=5,c=-1 18 30 a=3, b=30,c=-1
Explanation:	Explanation: Apply the rule for increment or decrement operators.

4.	public class Test {
	public static void main(String[] args) {
	System.*out*.print(011+ 1.94 + "C" + "S");
	}
	}

Answer:	10.94CS
Explanation:	011 is octal so first converted to decimal so it is 9. Then 9+1.94=10.94 then concatenation CS. So 10.94CS
5.	public class Test {

	public static void main(String[] args) {
	System.*out*.println(2+3+"bc"+'c'+'a');
	System.*out*.println('c'+'a'+2+3+"bc");
	System.*out*.println("bc"+'c'+'a'+2+3);
	System.*out*.println("bc"+('c'+'a')+(2)+3);
	}
	}

Answer:	5bcca 201bc bcca23 bc19623
Explanation:	: Apply the rule of concatenation and ASCII value of characters.

6.	public class Test {
	public static void main(String[] args) {
	int x = - 4;
	System.out.println(x>>1);
	int y = 4;
	System.out.println(y>>1);
	}
	}

Answer:	-2 2
Explanation:	-4 binary before right shift 11111111111111111111111111111100 -2 after right shift binary is 11111111111111111111111111111110 4 binary before right shift 100 2 after right shift binary is 10

7.	public class Test {
	public static void main(String[] args) {
	System.out.println(10 + 15 + "Hello");
	System.out.println("Hello" + 10 + 15);
	}
	}

Answer:	25Hello Hello1015
Explanation:	In the given expressions 10 + 15 + "Hello" and "Hello" + 10 + 15 , there are two + operators, so we check for associatively. The + operator is left to right. So the first expression is evaluated as (10 + 15) + “Hello” and second expression is evaluated as (“Hello” + 10) + 15.

8.	public class Test {
	public static void main(String[] args) {
	byte b = 10;
	b = b + 10;
	System.out.println(b);
	}

	}
Answer:	incompatible types: possible lossy conversion from int to byte
Explanation:	Normal assignment operator are used. Here assignment of an int (b+1=20) value to byte variable (i.e. b) is done for which the result is in compile time error. So type-casting (b=(byte)(b+10)) is done to get the result.

9.	public class Test {
	public static void main(String[] args) {
	int i = 4;
	int j = 21;
	int k = ++i * 7 + 2 - j--;
	System.out.println("k = " + k);
	}
	}

Answer:	16
Explanation:	k=5*7+2-21 and the evaluation will done by left to right. k=16.

10.	public class Test {
	public static void main(String[] args) {
	int a = 2;
	int b = 3;
	int result = a && b;
	System.out.println(result);
	}
	}

Answer:	Compilation error
Explanation:	The operator && is undefined for the argument type(s) int, int

11.	public class Test {
	public static void main(String[] args) {
	int x=-5;
	System.*out*.println(~x);
	}
	}

Answer:	4
Explanation:	5=0000-----0101 1’s Complement(5)=1111-----1010 -5=2’s Complement(5)=1111-----1011 1’s Complement(-5)=0000-----0100 =4

12.	public class Test {
	public static void main(String[] args) {
	int x=Integer.*MAX_VALUE*;
	System.*out*.println(x>>28);
	}
	}

Answer:	7
Explanation:	x=0111-----1111 x>>28=0000-----0111 =7

13.	public class Test {
	public static void main(String[] args) {
	int x=10,y=5;
	System.*out*.println(x++^++y|(x=y)&101);
	}
	}

Answer:	12
Explanation:	Use operator precedence table

14.	public class Test {
	public static void main(String[] args) {
	int x=-4,y=4;
	System.*out*.println((x>>>30)+" "+(x>>30)+" "+(y>>1));
	}
	}

Answer:	3 -1 2
Explanation:	-4=1111-----1100 -4>>>30=0000-----0011 =3 -4=1111-----1100 -4>>30=1111-----1111 (negative number so preserve sign (-)) 1’s Comp(-4>>30)=0000-----0000 2’s Comp(-4>>30)=0000—---0001 =1 Answer=-1

15.	public class Test {
	public static void main(String[] args) {
	int x=5;
	int y=x++ + ++x + ++x;
	int z=--y + x++ + y++;
	int p=z++ - (z%10) + (p=z);
	System.*out*.println(x+" "+y+" "+z+" "+p);
	}
	}

Answer:	9 20 47 86
Explanation:	Use operator precedence table.

16.	public class OperatorEx1 {
	public static void main(String args[]){
	int x=10;
	System.out.println(x++);
	System.out.println(++x);
	System.out.println(x--);
	System.out.println(--x);
	}
	}

Answer:	10 12 12 10
Explanation:	

17.	public class OperatorEx2 {
	public static void main(String args[]){
	int a=10;
	int b=10;
	System.out.println(a++ + ++a);//10+12=22
	System.out.println(b++ + b++);//10+11=21
	}
	}

Answer:	22 21
Explanation:	

18.	public class OperatorEx3 {
	public static void main(String args[]){
	System.out.println(10<<2);
	System.out.println(10<<3);
	System.out.println(20<<2);
	System.out.println(15<<4);
	}
	}

Answer:	40 80 80 240
Explanation:	

19.	public class OperatorEx4 {
	public static void main(String args[]){
	System.out.println(10>>2);
	System.out.println(20>>2);
	System.out.println(20>>3);
	}
	}

Answer:	2

	5 2
Explanation:	

20.	public class OperatorEx5{
	public static void main(String args[]){
	int a=10;
	int b=5;
	int c=20;
	System.out.println(a < b && a < c);
	System.out.println(a < b & a < c);
	}
	}

Answer:	false false
Explanation:	

21.	public class OperatorEx6{
	public static void main(String args[]){
	int a=10;
	int b=5;
	int c=20;
	System.out.println(a < b&&a++ < c);
	System.out.println(a);
	System.out.println(a < b&a++ < c);
	System.out.println(a);
	}
	}

Answer:	false 10 false 11
Explanation:	

22.	public class OperatorEx7{
	public static void main(String args[]){
	int a=10;
	int b=6;
	int c=30;
	System.out.println(a > b || a < c);
	System.out.println(a > b | a < c);
	System.out.println(a > b || a++ < c);
	System.out.println(a);
	System.out.println(a > b | a ++ < c);
	System.out.println(a);
	}
	}

Answer:	true true true 10 true

	11
Explanation:	

23.	public class Test{
	public static void main(String args[]){
	int a=4;
	int b=5;
	int x=(a++ < b)?a:b;//5 : 5
	int y=a+b-x;
	System.out.println("x="+x);
	System.out.println("y="+y);
	}
	}

Answer:	x=5 y=5
Explanation:	

24.	public class OperatorEx9{
	public static void main(String[] args){
	int a=10;
	a+=3;
	System.out.println(a);
	a-=4;
	System.out.println(a);
	a*=2;
	System.out.println(a);
	a/=2;
	System.out.println(a);
	}
	}

Answer:	13 9 18 9
Explanation:	

25.	public class IntegerConversion{
	public static void main(String args[]){
	long l = 55;
	int i = 44;
	short s = 33;
	byte b = 22;
	i = (int) l;
	s = (short) i;
	b = (byte) s;
	System.out.println("l = " + l);
	System.out.println("i = " + i);
	System.out.println("s = " + s);
	System.out.println("b = " + b);
	}
	}

Answer:	l = 55

	i = 55 s = 55 b = 55
Explanation:	

26.	public class Conversion2 {
	public static void main(String args[]) {
	int i = 132;
	short s = 15;
	byte b = (byte) i;
	int x = b + s;
	System.out.println("Value of x is " + x);
	}
	}

Answer:	Value of x is -109
Explanation:	

27.	public class IntegerGroupAddition{
	public static void main(String args[]){
	long l = 30;
	int i = 50;
	short s = 60;
	byte b = 70;
	byte sum = (byte)(l + i + s + b);
	System.out.println("Sum = " + sum);
	}
	}

Answer:	Sum = -46
Explanation:	

28.	Public class demo1{
	public static void main(String args[]){
	byte y=5,z=-y;
	System.out.println(~y);
	System.out.println(~z);
	y&= ~y;
	System.out.println(y);
	byte x = -1;
	System.out.println(x>>>6);
	byte a=-5,b=-6;
	System.out.println(a|b);
	}
	}

Answer:	-6 4 0 67108863 -5
Explanation:	X is stored using 8 bit 2's complement form. Binary representation of -1isall1s (11111111) The value of 'x>>>6'is 00000011

29.	Public class demo2{
	public static void main(String args[]) {
	System.out.println(2!=3 && (7>8 || 6>5));
	System.out.println(!(2!=3) && (7>8 || 6>5));
	System.out.println(3==3 && z>=10));
	System.out.println(2!=3 && (7>8 || 6>5));
	}
	}

Answer:	truefalsetruetrue
Explanation:	

30.	Public class demo3{
	public static void main(String args[]) {
	int v=10;
	System.out.println(v%=3*4);int x=11;
	System.out.println(-x- -); System.out.println(x);
	x = -x- -;
	System.out.println(x); int y = -x- -;
	System.out.println(x+""+y);
	}
	}

Answer:	10 -11 10 -10 -11 10
Explanation:	

31.	Public class demo4{
	public static void main(String args[]) {
	int x=-11;
	System.out.println(x%2);
	System.out.println(x/2);
	}
	}

Answer:	-1 -5
Explanation:	

32.	FIND Errors
	
	Public class demo5{
	public static voidmain(String args[]) {
	int 1stnum=10,nu-m2=20,3rd num=40;
	System.out.println("/"hello/"");
	byte b=128; float c=2.1; charc='a'; char cc=20;
	System.out.println(cc);
	}
	}

Answer:	errors
Explanation:	1stnum, nu-m2 and3rd numare not valid variable names

	Instead of /"hello/"wehave to write \"hello\" For bytetype variable the maximum value is 127 so it can’t store value larger than 127 c is float type. It can’t store adouble value. In the statementcharc=20 there is an error. The namec has been used again for declaring acharacter variable. In the lasttwo statements there is noerror.

33.	public class Test
	{
	public static void main(String[] args)
	{
	int a = 10;
	System.out.println(a++++);
	}
	}

Answer:	Compilation Error
Explanation:	It is evaluated as 10++; variable is required to perform ++ operator. Performing ++ on 10 is compilation error.

34.	public class Test
	{
	public static void main(String[] args)
	{
	int a=2;
	int b=4;
	System.out.println(“value of a XOR B:”+(a^b));
	}
	}

Answer:	value of a XOR B in Java : 6
Explanation:	0010^0100=6

35.	public class Test
	{
	public static void main(String[] args)
	{
	int a = 10;
	
	if(++a==11 || ++a==12)
	++a;
	System.out.println(a);
	}
	}

Answer:	12

Explanation:	11==11||11==12 12 in logical or if first condition is true it will not check the second condition so first a will be 11 and then 12

36.	public class Test
	{
	public static void main(String s[])
	{
	int a, b, result;
	a=10; b=20;
	result=(b>=a);
	System.out.println(result);
	}
	}

Answer:	Error: Incompatible type
Explanation:	The Expression result=(b>=a); here value of b is largest from a,True will return, and true (boolean) can not assign into integer variable.

37.	public class Test
	{
	public static void main (String[] args)
	{
	int x=20;
	String sup = (x < 15) ? "small" : (x < 22)?
	"tiny" : "huge";
	System.out.println(sup);
	}
	}

Answer:	tiny
Explanation:	This is an example of a nested ternary operator. The second evaluation (x < 22) is true, so the "tiny" value is assigned to sup.

38.	public class Alpha
	{
	public static void main(String args[])
	{
	int a=12+21*3-9/2;
	int b=14-32*4+175/8-3;
	
	boolean p=(++a>71&&--b<20);
	System.*out*.println(p);
	boolean p1=(b-- == -99 || a-- > 100);
	System.*out*.println(p1);

	} }
Answer:	true false
Explanation:	

39.	public class Alpha
	{
	public static void main(String[] args)
	{
	char a = 'A';
	System.*out*.println(++a +" "+ (int)a++);
	}
	}

Answer:	B 66
Explanation:	

40.	public class Alpha
	{
	public static void main(String[] args)
	{
	float x=5.3f;
	
	boolean p=(x==5.3);
	System.out.println(p);
	}
	}

Answer:	False
Explanation:	

41.	public class Alpha
	{
	public static void main(String[] args)
	{
	int temp = 9;
	int data = 8;
	System.*out*.println(temp & data);
	System.*out*.println(temp | data);
	System.*out*.println(temp ^ data);
	}
	}

Answer:	8 9 1
Explanation:	
42.	public class Alpha { public static void main(String[] args)

	{
	double d1 = 123.456;
	double d2 = 12_3.4_5_6;
	double d3 = 12_3.4_56;
	System.*out*.println(d1);
	System.*out*.println(d2);
	System.*out*.println(d3);
	
	}
	}

Answer:	123.456 123.456 123.456
Explanation:	we can use '_'(under Score) Symbol between digits of numeric literals according to java naming conventions.

43.	public class Test1 {
	
	public static void main(String[] args) {
	int x = 7;
	int y = 4;
	x+=4/3+x--+y+++x+++y--;
	System.*out*.print("x ="+x);
	System.*out*.print("y ="+y);
	}
	}

Answer:	x =30 y =4
Explanation:	It can be seen that, in line no. 5 and 6 the variables x and y are initialized by the values 7 and 4 respectively. Now, the expression given in line no. 7 has the following operators: +=, /, +, post increment, and post decrement. Following the precedence table, among the available operators both the post increment and post decrement are having the highest precedence. However, post increment and post decrement are having the same precedence, so due to this reason we will look for the associatively property in this case. As we know that, the post increment and post decrement operators have the associatively from left to right. Applying this information to the aforementioned expression, we will be getting: x+ = 4=3 + 7 + 4 + 6 + 5. Next, the = operator will be executed and will result in x+ = 1 + 7 + 4 + 6 + 5. After this, the +

	operator will be executed, and it can be seen that as we have multiple + operators, following the associatively property, the execution will go from left to right. So, we can write x+ = 23. and finally + = operator will be executed. It can be written as x = x+23. The current value of x is 7 and y is 4, so finally 30 will be stored in x and 4 will be stored in y.

44.	public class Test2
	{
	
	public static void main(String[] args)
	{
	int a, b = 10;
	a = -b --;
	System.*out*.println("a ="+a);
	System.*out*.println("b ="+b);
	}
	}

Answer:	a =-10 b =9
Explanation:	In line no. 5 the variables a is declared and variable b is declared and initialized with value 10. The expression given in line no. 6 has the following operators: =, - (unary), and - (Post decrement). Following the precedence table, among the available operators the post decrement operator is having the highest precedence, then unary and then the assignment operator (=). Following this information, the above expression can be rewritten as a = -(b--)and results in a = -(10). So, -10 will be assigned to a and 9 will be assigned to b.
45.	Which of the following are the legal identi_ers: (a) int a; (b) int :b; (c) int ____2_w; (d) int e#; (e) int this_is_a_very_detailed_name_for_an_identifier; (f) int $c; (g) int -d; (h) int -$; (i) int .f ; (j) int 7g;
Answer:	Legal Identi_ers: a, c, d, e, and h. Illegal Identi_ers: b, f, g, i, and j.
Explanation:	As we know that the identifiers are the sequence of

	characters that consists of letters, digits, underscores (_), and dollar signs ($). An identifier must start with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.

46.	public class Test3 {
	
	public static void main(String[] args)
	{
	int i = 1;
	byte b = i;
	System.*out*.print("b ="+b);
	}
	}

Answer:	Error: possible loss of precision Test3.java:5: possible loss of precision found : int required: byte
Explanation:	To assign a variable of the int type to a variable of the short or byte type, explicit casting must be used.

47.	public class Test4
	{
	public static void main(String[] args)
	{
	int a = 4, b=2;
	a*=a/b;
	System.*out*.print("a ="+a);
	System.*out*.print("b ="+b);
	}
	}

Answer:	a =8 b =2

48.	public class Alpha
	{
	public static void main(String[] args)
	{
	int x = 5 ;
	x = x << 3 + 2 ;
	System.out.println(“ x = “ + x);
	}
	}

Answer:	160
Explanation:	
49.	public class Alpha {

	public static void main(String[] args)
	{
	int x = 5 ;
	boolean r = x <2 && ++x > 4;
	System.out.println(" r = " + r + " x = "
	+ x);
	}
	}

Answer:	*false 5*
Explanation:	
50.	In which format -ve numbers are represented in computer memory ? a) 1's Complement format b) 2' Complement format c) Original binary equivalent of the number d) none of the above
Answer:	b) 2' Complement format
Explanation:	

51.	public class increment
	{
	public static void main(String args[])
	{
	double var1 = 1+5;
	double var2 = var1/4;
	int var3 = 1+5;
	int var4 = var3/4;
	
	System.out.print(var2 + " "+ var4);
	}
	}

Answer:	*1.5 1*
Explanation:	

52.	public class p1 {
	
	public static void main(String[] args)
	{
	
	int a=10, b=9;
	boolean k;
	k=(a<b) && (++b==a);
	System.*out*.println(b);

	`}` `}`
Answer:	9
Explation:	Because first expression is false so second expression will not not evaluate that is called concept of short circuit.

53.	`public class p2 {`
	
	`public static void main(String[] args)`
	`{`
	
	`final int a = 10;`
	`int b = ++a;`
	`System.out.println(b);`
	
	`}`
	
	`}`

Answer:	Error
Explanation:	Final value cannot be changed

54.	`public class p3 {`
	
	`public static void main(String[] args)`
	`{`
	`System.out.println( (10|5)+"-"+ (10|6));`
	
	`}`
	
	`}`

Answer:	15-14
Explanation:	Using Binary (1010|0101 = 1111 = 15) and 10|6 = Using Binary (1010|0110 = 1110 = 14

55.	`public class p4 {`
	
	`public static void main(String[] args)`
	`{`
	
	`String s1 = "ITER";`
	`String s2 = "ITER";`
	`System.out.println("s1 == s2 is:" + s1 == s2);`
	`}`
	`}`

Answer:	false
Explation:	The output is “false” because in java + operator precedence is more than == operator. So the given expression will be evaluated to “s1 == s2 is:ITER” == “ITER” i.e false.

56.	public class p5 {
	
	public static void main(String[] args)
	{
	int x = -1;
	System.*out*.println(x>>>29);
	System.*out*.println(x>>>30);
	System*out*.println(x>>>31);
	
	}
	}

Answer:	7 3 1
Explanation:	X is stores in 32 bits in form of 2’s complement. i.e all bits are 11111...111

57.	public class p6 {
	
	public static void main(String[] args)
	{
	byte x=127; // Line 5
	x= x << 3; // Line 6
	System.*out*.println(x);
	}
	}

Answer:	Error at Line 6: incompatible types: possible lossy conversion from int to byte
Explanation:	Because, the shift operation always assume integer variable as its operand, and, an integer variable can’t be stored in a byte variable.

58.	public class p7 {
	
	public static void main(String[] args)
	{
	int x=127, y=128;
	x= (x & 3) | y;
	System.*out*.println(x);
	}
	
	}

Answer:	131
Explation:	The value of x = 127 and y =128 are represented

	in Binary form. Then (x & 3) is computed and then bitwise-OR operation is performed with the result of (x & 3). Then, the binary resultant is converted to decimal and stored in x and then displayed.

59.	`public class p8 {`
	
	`public static void main(String[] args)`
	`{`
	`int x= 9, y=0;`
	`System.out.println((++x)==10 && (++y)==1);`
	`}`
	
	`}`

Answer:	`true`
Explanation:	Initially, the ++x is computed, i.e., 10 and ++y is computed, i.e., 1. Since, both the conditions are true, the answer is true.

60.	`public class p9 {`
	
	`public static void main(String[] args)`
	`{`
	`int x=127;  // Line 5`
	`x+= (x << 3);  // Line 6`
	`System.out.println(x);`
	`}`
	
	`}`

Answer:	`1143`
Explanation:	The expression at Line 6 will be expanded as follows: x =x + (x <<3). So, initially the x <<3 is computed and then added with x to obtain the final resultant. This resultant will be again convert from binary to decimal and stored in x for displaying.

61.	`public class p10 {`
	
	`public static void main(String[] args)`
	`{`
	`int x=12, y=7, z=9;  // Line 5`
	`z= (x<y)?  (x > z ? z: x) : (y < z ? z: y);`
	`System.out.println(z);`
	`}`
	`}`

Answer:	`9`
Explation:	The condition x < y is true, this means, the

	expression (x > z ? z : x) will execute. Again, the condition x > z is true, which means, z = z. Therefore, z will display 9.

62.	`public class p11 {`
	
	`public static void main(String[] args)`
	`{`
	`int ++a=100;`
	`System.out.println(a++);`
	`}`
	
	`}`

Answer:	Syntax Error
Explanation:	Compiler displays error as ++a is not a valid identifier

63.	`public class p12 {`
	
	`public static void main(String[] args)`
	`{`
	`int x = 100;`
	`double y = 100.1;`
	`boolean b = (x=y);  //Line 7`
	`System.out.println(b);`
	`}`
	
	`}`

Answer:	Compilation fails (Syntax error)
Explanation:	The code will not compile because in line 7, the line will work only if we use(x==y) in the line. The==operator compares values to produce a boolean, whereas the=operator assigns a value to variables.
64.	With x = 0, which of the following are legal lines of Java code for changing the value of x to 1? 1. x++; 2. x=x+1; 3. x+=1; 4. x=+1;
Answer:	All lines are legal
Explation:	Operator ++ increases value of variable by 1. x = x + 1 can also be written in shorthand form as x += 1. Also x =+ 1 will set the value of x to 1.

65.	`public class p14 {`
	
	`public static void main(String[] args)`

	{
	
	int x;
	System.*out*.println(x);
	}
	}

Answer:	compiler error
Explanation:	Unlike class members, local variables of methods must be assigned a value to before they are accessed, or it is a compile error.

66.	public class p15 {
	
	public static void main(String[] args)
	{
	
	double a, b, c;
	a = 3.0/0;
	b = 0/4.0;
	c=0/0.0;
	System.*out*.println(a);
	System.*out*.println(b);
	System.*out*.println(c);
	}
	}

Answer:	Infinity 0.0 NaN
Explanation:	For floating point literals, we have constant value to represent (3.0/0) infinity either positive or negative and also have NaN (not a number for undefined like 0/0.0), but for the integral type, we don’t have any constant that’s why we get an arithmetic exception.

67.	public class p16 {
	
	public static void main(String[] args)
	{
	
	// the line below this gives an output
	// \u000d System.*out*.println("comment executed");
	
	}
	
	}

Answer:	comment executed
Explation:	The reason for this is that the Java compiler parses

	the unicode character \u000d as a new line and gets transformed into:
	`public class prog16 {`
	`  public static void main(String[] args)`
	`  {`
	`  // the line below this gives an output`
	`  // \u000d`
	`  System.out.println("comment executed");`
	`  }`
	
	`}`

68.	`public class p17 {`
	
	`  public static void main(String[] args)`
	`  {`
	
	`  int $_ = 5;`
	`  System.out.println($_);`
	`  }`
	
	`}`

Answer:	5
Explanation:	It looks like $ will cause an error, but it won't. In java, identifier rule says, identifier can start with any alphabet or underscore ("_") or dollar ("$"). So answer is 5.

69.	`public class p18 {`
	
	`  public static void main(String[] args)`
	`  {`
	
	`  String s1 = "abc";`
	`  String s2 = s1;`
	`  s1 += "d";`
	`  System.out.println(s1+" "+s2+" "+(s1 == s2));`
	`  }`
	`}`

Answer:	abcd abc false
Explanation:	In Java, String is immutable. So string s2 and s1 both pointing to the same string abc. And, after making the changes the string s1 points to abcd and s2 points to abc, hence false.

70.	`public class p19 {`
	
	`public static void main(String[] args)`
	`{`
	`int a = 5;`
	`System.out.println(a>>33);`
	`}`
	`}`

Answer:	2
Explation:	Modulo reduction is applied to the right hand operand based on the width of the left hand operand. given: l (shift operation) r If l is an int (32 bits) r is reduced to (r % 32). In your example, (33 % 32) == 1.

71.	`public class p20 {`
	
	`public static void main(String[] args)`
	`{`
	`int x = 07;`
	`int y = 08;`
	`System.out.println("" + x + y);`
	`}`
	`}`

Answer:	error
Explanation:	Any number beginning with zero is treated as an octal number (which is 0-7).