**Precedence and associativity of operators related tricky questions:**

int x=6;

x=++x;

System.out.println(“The value of x:”+x);

Now, ++ (increment operator) (precedence rank 2) has more priority than = (assignment operator precedence rank =14)

expression is checked form left to right to find high precedence operators

x=++x;

x=++6; (left to right searching. Now, this is pre-increment operator (associativity right to left, but there is no need to focus on associativity. Since, associativity only comes into scenario when two or more operators of same precedence are found in the same expression. Then associativity order decides which one to evaluate first)

++6 (first value is incremented then it is evaluated)

x=7 (finally) (after pre-increment operator is performed, the value returned is assigned to the x

int x=6;

x+=++x;

System.out.println(“The value of x:”+x);

in x+=++x two operators are used. “+=” (which has precedence rank 14) and “++” (increment operator which has precedence operator is evaluated)

x+=++x;

Now, first step

pre-increment operator performs its operation. first x’s value is incremented, then it is evaluated. x+=7; so, the value returned is 7.

Now, remember x+=a… += is a compound assignment operator where the right side value of assignment operator is added to the left side value of the assignment operator, then it is assigned to the left side variable of the assignment operator.

Now, the left side value of assignment operator is 6. it is added to 7. then 13(the result) is assigned to x. (so, x’s value is 13)

the increment operator is evaluated. Now, since it is pre-incremented first the value of x

int x=6;

x=x++;

System.out.println(“The value of x:”+x);

Now, among the assignment operator (precedence rank 14) and ++(increment rank=2), increment operator has higher precedence.

now, suppose, x’s original value is stored in X resister. now, it is copied (in proper terms, we call it fetching) to a temporary resister (say r1). Now, post-incremented operator first evaluates the value of x. then it is in incremented. so, the value returned is 6(since, first it evaluates the value, then increment it)

int x=6;

x+=x++;

System.out.println(“The value of x:”+x);

Now, remember, x+=a means x=x+a (i.e. the left side value of the assignment operator (which is a in this case) is added to the left side value of assignment operator.

Now, first post increment operation is performed. now, in post increment operation, first the value of the variable is evaluated, then it is incremented. so, the value returned after the post-increment operation is 6. (because, first it is evaluated) and it is added to the left side value of the assignment operator which is 6. So, the value which will be assigned to x finally is 12.

int x=6;

x+=++x+x++;

System.out.println(“The value of x:”+x);

Now, both pre-increment and post increment operator has same precedence. But their associativity order is from right to left. So, first post-increment operation is performed.

So, x+=++x+x++

will be treated as

x+=++x+(x++)

now, here, x’s value is first evaluated then incremented

x+=++x+(6) with x’s value as 7 (now) (but remember, the left side value of x remains unchanged

until an assignment operation is performed)

x+=(++x)+(6) (since, among the left operators, ++ operator has higher precedence)

x+=(8)+(6) since in the case of pre-increment, first the value is incremented then evaluated.

x+=14 (among, the left operators, ‘+’ addition operator has higher precedence than +=)

now, finally, the += operator will perform its own operation.

Now, it adds the right side value of the assignment operator with the left side value of the assignment operator. (Now, in left side of the assignment operator, x’s value is still 6.

So, ultimate value is 20 which is assigned to x.