https://www.geeksforgeeks.org/quizzes/operators-gq/?page=1

#include <stdio.h>

int main()

{

    int i = 1, 2, 3;

    printf("%d", i);

    return 0;

}

1

3

Garbage value

* Right

Compile time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-2/)

Question 1 ‒ **Explanation**

Comma acts as a separator here. The compiler creates an integer variable and initializes it with 1. The compiler fails to create integer variable 2 because 2 is not a valid identifier.

Compiler đang mong đợi biến để gán cho 2 và 3 không thể để 2 là biến, vì không khai báo biến mà gán giá trị nên bị lỗi

**Question 2**

#include <stdio.h>

int main()

{

    int i = (1, 2, 3);

    printf("%d", i);

    return 0;

}

1

* Right

3

Garbage value

Compile time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-3/)

Question 2 ‒ **Explanation**

The bracket operator has higher precedence than assignment operator. The expression within bracket operator is evaluated from left to right but it is always the result of the last expression which gets assigned.

**Question 3**

#include <stdio.h>

int main()

{

    int i;

    i = 1, 2, 3;

    printf("%d", i);

    return 0;

}

* Right

1

3

Garbage value

Compile time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-4/)

Question 3 ‒ **Explanation**

Comma acts as an operator. The assignment operator has higher precedence than comma operator. So, the expression is considered as (i = 1), 2, 3 and 1 gets assigned to variable i.

#include <stdio.h>

int foo(int\* a, int\* b)

{

int sum = \*a + \*b;

\*b = \*a;

return \*a = sum - \*b;

}

int main()

{

int i = 0, j = 1, k = 2, l;

l = i++ || foo(&j, &k);

printf("%d %d %d %d", i, j, k, l);

return 0;

}

Câu 4 – Giải thích

Điều khiển trong logic OR chỉ chuyển sang biểu thức thứ hai nếu biểu thức đầu tiên dẫn đến FALSE. Hàm foo() được gọi vì

tôi++

trả về 0(tăng sau) sau khi tăng giá trị của i lên 1. Hàm foo() thực sự hoán đổi giá trị của hai biến và trả về giá trị của tham số thứ hai. Vì vậy, giá trị của biến j và k được trao đổi và biểu thức OR được đánh giá là TRUE.

**Question 5**

C

#include *<stdio.h>*

int main()

{

int i = 5, j = 10, k = 15;

printf("%d ", **sizeof**(k /= i + j));

printf("%d", k);

**return** 0;

}

Assume size of an integer as 4 bytes. What is the output of above program?

* Wrong

4 1

* Right

4 15

2 1

Compile-time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-7/)

Question 5 ‒ **Explanation**

The main theme of the program lies here: **sizeof(k /= i + j)**. An expression doesn't get evaluated inside **sizeof** operator. **sizeof** operator returns sizeof(int) because the result of expression will be an integer. As the expression is not evaluated, value of **k** will not be changed.

k/=i trước sau đó cộng j nên cuối cùng k ra 13, k=k/I +j , nhưng phép toán trong sizeof không thực thi nên k vẫn bằng 15

**Question 6**

C

#include *<stdio.h>*

int main()

{

*//Assume sizeof character is 1 byte and sizeof integer is 4 bytes*

printf("%d", **sizeof**(printf("GeeksQuiz")));

**return** 0;

}

GeeksQuiz4

4GeeksQuiz

GeeksQuiz9

* Right

4

* Wrong

Compile-time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-8/)

Question 6 ‒ **Explanation**

An expression doesn't get evaluated inside **sizeof** operator. **GeeksQuiz** will not be printed. **printf** returns the number of characters to be printed i.e. 9 which is an integer value. **sizeof** operator returns sizeof(int).

**Question 7**

Output of following program?

C

#include *<stdio.h>*

int f1() { printf ("Geeks"); **return** 1;}

int f2() { printf ("Quiz"); **return** 1;}

int main()

{

int p = f1() + f2();

**return** 0;

}

GeeksQuiz

QuizGeeks

* Right

Compiler Dependent

Compiler Error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-9/)

Question 7 ‒ **Explanation**

The operator ‘+’ doesn't have a standard defined order of evaluation for its operands. Either f1() or f2() may be executed first. So a compiler may choose to output either “GeeksQuiz” or “QuizGeeks”.

**Question 8**

What is the output of following program?

C

#include *<stdio.h>*

int main()

{

int a = 1;

int b = 1;

int c = a || --b;

int d = a-- && --b;

printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d);

**return** 0;

}

a = 0, b = 1, c = 1, d = 0

* Right

a = 0, b = 0, c = 1, d = 0

* Wrong

a = 1, b = 1, c = 1, d = 1

a = 0, b = 0, c = 0, d = 0

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-10/)

Question 8 ‒ **Explanation**

Let us understand the execution line by line. Initial values of a and b are 1.

// Since a is 1, the expression --b

// is not executed because

// of the short-circuit property

// of logical or operator

// So c becomes 1, a and b remain 1

int c = a || --b;

// The post decrement operator --

// returns the old value in current expression

// and then updates the value. So the

// value of expression a-- is 1. Since the

// first operand of logical and is 1,

// shortcircuiting doesn't happen here. So

// the expression --b is executed and --b

// returns 0 because it is pre-increment.

// The values of a and b become 0, and

// the value of d also becomes 0.

int d = a-- && --b;

**Question 9**

C

#include *<stdio.h>*

int main()

{

int a = 10, b = 20, c = 30;

**if** (c > b > a)

printf("TRUE");

**else**

printf("FALSE");

**return** 0;

}

TRUE

* Right

FALSE

* Wrong

Compiler Error

Output is compiler dependent

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-11/)

Question 9 ‒ **Explanation**

Let us consider the condition inside the if statement. Since there are two greater than (>) operators in the expression “c > b > a”, associativity of > is considered. Associativity of > is left to right. So, expression c > b > a is evaluated as ( (c > b) > a ). And since the (c > b) is being the relational operator it will return 1 if True otherwise 0 is if False. So here the value returned is 1 and then it is compared to the a. so now, the statement becomes, (1 > a), which is false, so the answer, return is 0, therefore, else part is executed.

**Question 10**

C

#include*<stdio.h>*

int main()

{

char \*s[] = { "knowledge","is","power"};

char \*\*p;

p = s;

printf("%s ", ++\*p);

printf("%s ", \*p++);

printf("%s ", ++\*p);

**return** 0;

}

is power

* Right

nowledge nowledge s

is ower

nowledge knowledge is

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-12/)

Question 10 ‒ **Explanation**

Let us consider the expression ++\*p in first printf(). Since precedence of prefix ++ and \* is same, associativity comes into picture. \*p is evaluated first because both prefix ++ and \* are right to left associative. When we increment \*p by 1, it starts pointing to second character of “knowledge”. Therefore, the first printf statement prints “nowledge”. Let us consider the expression \*p++ in second printf() . Since precedence of postfix ++ is higher than \*, p++ is evaluated first. And since it’s a postfix ++, old value of p is used in this expression. Therefore, second printf statement prints “nowledge”. In third printf statement, the new value of p (updated by second printf) is used, and third printf() prints “s”.

**Question 11**

C

#include*<stdio.h>*

int main(void)

{

int a = 1;

int b = 0;

b = a++ + a++;

printf("%d %d",a,b);

**return** 0;

}

3 6

* Right

Compiler Dependent

3 4

3 3

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-13/)

Question 11 ‒ **Explanation**

It is compiler dependent as different compilers takes different bytes for integers.

**Question 12**

Predict the output of following program. Assume that the characters are represented using ASCII Values.

C

#include *<stdio.h>*

#define VAL 32

int main()

{

char arr[] = "geeksquiz";

\*(arr + 0) &= ~VAL;

\*(arr + 5) &= ~VAL;

printf("%s", arr);

**return** 0;

}

* Right

GeeksQuiz

geeksQuiz

* Wrong

Geeksquiz

geeksquiz

Garbage eeks Garbage uiz

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-15/)

Question 12 ‒ **Explanation**

The crux of the question lies in the statement: *\*(arr + 5) &= ~VAL;* This statement subtracts 32 from the ascii value of a lower case character and thus converts it to upper case. This is another way to convert an alphabet to upper case by resetting its bit positioned at value 32 i.e. 5th bit from the LSB(assuming LSB bit at position 0).

**Question 13**

Predict the output of the below program:

C

#include *<stdio.h>*

int main()

{

printf("%d", 1 << 2 + 3 << 4);

**return** 0;

}

* Wrong

112

52

* Right

512

0

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-16/)

Question 13 ‒ **Explanation**

The main logic behind the program is the [precedence](http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B#Operator_precedence) and associativity of the operators. The addition(+) operator has higher precedence than shift(<<) operator. So, the expression boils down to *1 << (2 + 3) << 4* which in turn reduces to (1 << 5) << 4 as the shift operator has left-to-right associativity.

**Question 14**

Which of the following can have different meaning in different contexts?

&

\*

* Right

Both of the above

There are no such operators in C

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-17/)

Question 14 ‒ **Explanation**

'&' can be used to get address of a variable and can also be used to do bitwise and operation. Similarly '\*' can be used to get value at an address and can also be used to multiplication.

**Question 15**

In C, two integers can be swapped using minimum

* Right

0 extra variable

* Wrong

1 extra variable

2 extra variable

4 extra variable

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-18/)

Question 15 ‒ **Explanation**

We can swap two variables without any extra variable using bitwise XOR operator '^'. Let X and Y be two variables to be swapped. Following steps swap X and Y.

X = X ^ Y;

Y = X ^ Y;

X = X ^ Y;

See <http://en.wikipedia.org/wiki/XOR_swap_algorithm>

**Question 16**

What does the following statement do?

C

x = x | 1 << n;

Sets x as 2n

* Right

Sets (n+1)th bit of x

* Wrong

Toggles (n+1)th bit of x

Unsets (n+1)th bit of x

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-19/)

Question 16 ‒ **Explanation**

Let n be 3, the value of expression 1 << n would be 8 (In binary 00001000). If we do bitwise or of binary 00001000 with any number x would set its (n+1)th bit from left side.

**Question 17**

Predict the output of following C program

C

#include *<stdio.h>*

int main()

{

int i = 0;

**do**

{

printf("GeeqsQuiz ");

i = i++;

}

**while** (i < 5);

**return** 0;

}

GeeqsQuiz GeeqsQuiz GeeqsQuiz GeeqsQuiz GeeqsQuiz

* Wrong

Infinite time GeeksQuiz

* Right

Undefined Behavior

None of the above

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-20/)

Question 17 ‒ **Explanation**

The below statement causes undefined behavior.

i = i++;

**Question 18**

Assume that the size of an integer is 4 bytes, predict the output of following program.

C

#include *<stdio.h>*

int main()

{

int i = 12;

int j = **sizeof**(i++);

printf("%d %d", i, j);

**return** 0;

}

* Right

12 4

* Wrong

13 4

Compiler Error

0 4

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-21/)

Question 18 ‒ **Explanation**

The expressions written inside sizeof are not evaluated, so i++ is not performed.

**Question 19**

C

#include*<stdio.h>*

int main()

{

int a = 2,b = 5;

a = a^b;

b = b^a;

printf("%d %d",a,b);

**return** 0;

}

5 2

2 5

* Wrong

7 7

* Right

7 2

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-22/)

Question 19 ‒ **Explanation**

^ is bitwise xor operator. a = 2 (10) b = 5 (101) a = a^b (10 ^ 101) = 7(111) b = a^b (111 ^ 101) = 2(10)

**Question 20**

Predict the output of following program?

C

#include *<stdio.h>*

int main() {

int x = 10;

int y = 20;

x += (y += 10);

printf("%d %d", x, y);

**return** 0;

}

40 20

* Right

40 30

30 30

30 40

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-23/)

Question 20 ‒ **Explanation**

The main statement in question is "x += y += 10". Since there are two += operators in the statement, associativity comes into the picture. Associativity of compound assignment operators is right to left, so the expression is evaluated as x += (y += 10).

**Question 21**

Output of following C code will be?

C

#include *<stdio.h>*

int main() {

int x = 10;

int y = (x++, x++, x++);

printf("%d %d**\n**", x, y);

**return** 0;

}

* Right

13 12

* Wrong

13 13

10 10

Compiler Dependent

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-24/)

Question 21 ‒ **Explanation**

Option A is correct.

**Question 22**

Output of following C code will be?

C

#include *<stdio.h>*

int main() {

int y = 0;

int x = (y != 0);

printf("%d", x);

**return** 0;

}

* Right

0

* Wrong

1

A bog negative Number

Compiler Error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-25/)

Question 22 ‒ **Explanation**

0 is the output of the following C code.

**Question 23**

Output of following C code will be?

C

#include *<stdio.h>*

int main()

{

int a = 0;

int b;

a = (a == (a == 1));

printf("%d", a);

**return** 0;

}

0

* Right

1

Big negative number

-1

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-26/)

Question 23 ‒ **Explanation**

We need to figure out value of "(a == (a == 1))"   
  
(a == 1) returns false as a is initialized as 0. So in outer bracket, false is compared with a. Since a is 0, result of of outer bracket becomes true.   
  
The important thing to note is, in C, when a boolean value is compared or assigned to an integer value, false is considered as 0 and true is considered as 1.

**Question 24**

Output of following C code will be?

C

#include *<stdio.h>*

#include *<stdlib.h>*

int top = 0;

char fun1()

{

char a[] = {'a', 'b', 'c', '(', 'd'};

**return** a[top++];

}

int main()

{

char b[10];

char ch2;

int i = 0;

**while** ((ch2 = fun1()) != '(')

{

b[i++] = ch2;

}

b[i] = '\0'; *// Add null-terminating character to mark the end of the string*

printf("%s", b);

**return** 0;

}

abc(

* Right

abc

3 special characters with ASCII value 1

Empty String

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-27/)

Question 24 ‒ **Explanation**

This program prints "abc" as the output.

**Question 25**

In the context of modulo operation (i.e. remainder on division) for floating point (say 2.1 and 1.1), pick the best statement.

For floating point, modulo operation isn't defined and that's why modulo can't be found.

* Wrong

(2.1 % 1.1) is the result of modulo operation.

* Right

fmod(2.1,1.1) is the result of module operation.

((int)2.1) % ((int)1.1) is the result of modulo operation.

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-c-quiz-102-question-4/)

Question 25 ‒ **Explanation**

% works on integer types only not for floating types. Typecasting to integer type might approximate the intended result but it won't produce the correct result. Basically the

*fmod(x,y*

) function returns the value x − ny, for some integer n such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

*fmod()*

is declared in "math.h" and its prototype is "

*double fmod(double x, double y)*

". For

*float*

and

*long double*

also, modulo has been implemented in math.h library through

*fmodf()*

and

*fmodl()*

.

**Question 26**

For a given integer, which of the following operators can be used to “set” and “reset” a particular bit respectively?

* Right

| and &

* Wrong

&& and ||

& and |

|| and &&

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-c-quiz-103-question-1/)

Question 26 ‒ **Explanation**

Bitwise operator | can be used to “set” a particular bit while bitwise operator & can be used to “reset” a particular bit. Please note that && and || are logical operators which evaluate their operands to logical TRUE or FALSE. It should be noted that bitwise operator & can be used for checking a particular bit also i.e. whether a bit is set or not. So correct answer it A.

**Question 27**

Suppose a, b, c and d are int variables. For ternary operator in C ( ? : ), pick the best statement.

a>b ? : ; is valid statement i.e. 2nd and 3rd operands can be empty and they are implicitly replaced with non-zero value at run-time.

* Wrong

a>b ? c=10 : d=10; is valid statement. Based on the value of a and b, either c or d gets assigned the value of 10.

* Right

a>b ? (c=10,d=20) : (c=20,d=10); is valid statement. Based on the value of a and b, either c=10,d=20 gets executed or c=20,d=10 gets executed.

All of the above are valid statements for ternary operator.

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-c-quiz-107-question-1/)

Question 27 ‒ **Explanation**

For ternary operator, both 2nd and 3rd operands are necessary. So A) isn’t correct. As per operator precedence, ternary operator has higher precedence over assignment operator. So B) isn’t correct.

**Question 28**

The below program would give compile error because comma has been used after foo(). Instead, semi-colon should be used i.e. the way it has been used after bar(). That's why if we use semi-colon after foo(), the program would compile and run successfully while printing "GeeksQuiz"

C

#include *<stdio.h>*

void foo(void)

{

printf("Geeks");

}

void bar(void)

{

printf("Quiz");

}

int main()

{

foo(), bar();

**return** 0;

}

* Wrong

TRUE

* Right

FALSE

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-c-quiz-110-question-4/)

Question 28 ‒ **Explanation**

Here, comma is acting as an operator instead of separator. For a comma operator in C, first left operand is evaluated and then right operand is evaluated. That's why foo() would be called followed by bar(). There's no issue with the given program. It'll compile and print "GeeksQuiz" without any modification itself.

**Question 29**

What does the following program do when the input is unsigned 16-bit integer?

#include

main( )

{

unsigned int num;

int i;

scanf (“%u”, &num);

for ( i = 0; i<16; i++)

{

printf (“%d”, (num << i & 1 << 15 ) ? 1:0);

}

}

It prints all even bits from num

* Wrong

It prints all odd bits from num

* Right

It prints binary equivalent of num

None of the above

[**Discuss it**](https://www.geeksforgeeks.org/questions/isro-isro-cs-2017-may-question-61/)

**Question 30**

Which of the following operators cannot be overloaded in C/C++ ?

Bitwise right shift assignment

* Wrong

Address of

Indirection

* Right

Structure reference

[**Discuss it**](https://www.geeksforgeeks.org/questions/ugc-net-ugc-net-cs-2017-jan-ii-question-14/)

Question 30 ‒ **Explanation**

Structure reference cannot be overloaded in C/C++. For detailed information Refer:[Functions that cannot be overloaded in C++](https://www.geeksforgeeks.org/function-overloading-in-c/) So, option (D) is correct.

**Question 31**

Write the output of the following C program

C

#include *<stdio.h>*

int main (void)

{

int shifty;

shifty = 0570;

shifty = shifty >>4;

shifty = shifty <<6;

printf("the value of shifty is %o",shifty);

}

the value of shifty is 15c0

* Wrong

the value of shifty is 4300

the value of shifty is 5700

* Right

the value of shifty is 2700

[**Discuss it**](https://www.geeksforgeeks.org/questions/isro-isro-cs-2014-question-32/)

Question 31 ‒ **Explanation**

Given, shifty = 0570; here shifty is a octal number as it starts with 0.

(0570)8 = (000 101 111 000)2

Right shift by 4 places:

shifty = shifty >>4;

shifty = (000 000 010 111)2

Left shift by 6 places:

shifty = shifty <<6

shifty = (010 111 000 000)2

= (2700)8

So, option (D) is correct.

**Question 32**

The following three 'C' language statements is equivalent to which single statement? y=y+1; z=x+y; x=x+1

z = x + y + 2;

* Right

z = (x++) + (++y);

* Wrong

z = (x++) + (y++);

z = (x++) + (++y) + 1;

[**Discuss it**](https://www.geeksforgeeks.org/questions/isro-isro-cs-2014-question-47/)

Question 32 ‒ **Explanation**

In 'C' language, there are two type of increment operators:

post increment:  
b = a++;  
b = a;  
a = a + 1.  
pre increment:  
b = ++a;  
a = a + 1  
b = a;

y=y+1; //++y  
z=x+y; //z = x + (++y)  
x=x+1 //z = (x++) + (++y)

So, option (B) is correct.

**Question 33**

Given i = 0, j = 1, k = –1 x = 0.5, y = 0.0 What is the output of the following expression in C language ? x \* y < i + j || k

-1

0

* Right

1

2

[**Discuss it**](https://www.geeksforgeeks.org/questions/ugc-net-ugc-net-cs-2016-july-ii-question-11/)

Question 33 ‒ **Explanation**

= x \* y < i + j || k

= 0.5 \* 0.0 < 0 + 1 || -1

= 0.0 < 0 + 1 || -1

= 0.0 < 1 || -1

= 1 || -1

= 1

So, option (C) is correct.

**Question 34**

Loop unrolling is a code optimization technique:

* Right

that avoids tests at every iteration of the loop.

that improves performance by decreasing the number of instructions in a basic block.

* Wrong

that exchanges inner loops with outer loops

that reorders operations to allow multiple computations to happen in parallel

[**Discuss it**](https://www.geeksforgeeks.org/questions/ugc-net-ugc-net-cs-2015-dec-ii-question-41/)

Question 34 ‒ **Explanation**

Loop unrolling is a code optimization technique that avoids tests at every iteration of the loop. It does not improves performance by decreasing the number of instructions in a basic block. Neither it exchanges inner loops with outer loops nor it reorders operations to allow multiple computations to happen in parallel. So, option (A) is correct.

**Question 35**

Output of following C++ code will be?

C++

#include *<iostream>*

**using** **namespace** **std**;

**class** **X**

{

**public**:

int x;

};

int main()

{

X a = {10};

X b = a;

cout << a.x << " " << b.x;

**return** 0;

}

Compiler Error

10 followed by Garbage Value

* Right

10 10

10 0

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-constructors-question-6/)

Question 35 ‒ **Explanation**

The following may look like an error, but it works fine. X a = {10}; Like structures, class objects can be initialized. The line "X b = a;" calls copy constructor and is same as "X b(a);". Please note that, if we don't write our own copy constructor, then compiler creates a default copy constructor which assigns data members one object to other object.

**Question 36**

Output of following program? 

C++

#include *<iostream>*

**using** **namespace** **std**;

**class** **Point** {

Point() { cout << "Constructor called"; }

};

int main()

{

Point t1;

**return** 0;

}

Runtime Error

None of these

* Wrong

Constructor called

* Right

Compiler Error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-constructors-question-3/)

Question 36 ‒ **Explanation**

By default all members of a class are private. Since no access specifier is there for Point(), it becomes private and it is called outside the class when t1 is constructed in main. 

**Question 37**

Given that x = 7.5, j = -1.0, n = 1.0, m = 2.0 the value of - - x + j == x>n> = m is:

* Right

0

1

2

* Wrong

3

[**Discuss it**](https://www.geeksforgeeks.org/questions/ugc-net-ugc-net-cs-2015-jun-ii-question-13/)

Question 37 ‒ **Explanation**

- - x + j == x > n >= m   
6.5 + (-1.0) == 6.5 > 1.0 >= 2.0   
5.5 == 1 > = 2.0   
3.5 == 0   
Which is not true, i.e. false(0).   
So, option (A) is correct. 

**Question 38**

If n has 3, then the statement a[++n]=n++;

assigns 3 to a[5]

assigns 4 to a[5]

* Wrong

assigns 4 to a[4]

* Right

what is assigned is compiler dependent

[**Discuss it**](https://www.geeksforgeeks.org/questions/isro-isro-cs-2015-question-68/)

**Question 39**

C

#include *"stdio.h"*

int main()

{

int x, y = 5, z = 5;

x = y == z;

printf("%d", x);

getchar();

**return** 0;

}

0

* Right

1

* Wrong

5

Compiler Error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-1/)

Question 39 ‒ **Explanation**

The crux of the question lies in the statement x = y==z. The operator == is executed before = because precedence of comparison operators (<=, >= and ==) is higher than assignment operator =.   
The result of a comparison operator is either 0 or 1 based on the comparison result. Since y is equal to z, value of the expression y == z becomes 1 and the value is assigned to x via the assignment operator.

**Question 40**

C

#include *<stdio.h>*

int main()

{

int i = 3;

printf("%d", (++i)++);

**return** 0;

}

What is the output of the above program?

3

4

* Wrong

5

* Right

Compile-time error

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-5/)

Question 40 ‒ **Explanation**

In C, prefix and postfix operators need l-value to perform operation and return r-value. The expression **(++i)++** when executed increments the value of variable i(i is a l-value) and returns r-value. The compiler generates the error(l-value required) when it tries to post-incremeny the value of a r-value.

**Question 41**

Which of the following is not a logical operator?

* Wrong

&&

!

||

* Right

|

[**Discuss it**](https://www.geeksforgeeks.org/questions/c-operators-question-14/)

Question 41 ‒ **Explanation**

**&&**- Logical AND   
**!**- Logical NOT   
**||**- Logical OR   
**|**- Bitwise OR(used in bitwise manipulations)