1. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=-3, j=2, k=0, m;
    m = ++i && ++j && ++k;
    printf("%d, %d, %d, %d\n", i, j, k, m);
    return 0;
}
```

A. -2, 3, 1, 1

B. 2, 3, 1, 2

C. 1, 2, 3, 1

D. 3, 3, 1, 2

Answer: Option **A**
Explanation:
Step 1: `int i=-3, j=2, k=0, m;` here variable `i, j, k, m` are declared as an integer type and variable `i, j, k` are initialized to -3, 2, 0 respectively.
Step 2: `m = ++i && ++j && ++k;`
becomes `m = -2 && 3 && 1;`
becomes `m = TRUE && TRUE;` Hence this statement becomes TRUE. So it returns '1'(one). Hence m=1.
Step 3: `printf("%d, %d, %d, %d\n", i, j, k, m);` In the previous step the value of i,j,k are increemented by '1'(one).

Hence the output is "-2, 3, 1, 1".

View Answer Discuss in Forum Workspace Report

---

2. Assuming, integer is 2 byte, What will be the output of the program?

```
#include<stdio.h>

int main()
{
    printf("%x\n", -2<<2);
    return 0;
}
```

A. ffff

B. 0

C. fff8

D. Error

Answer: Option **C**
Explanation:
The integer value 2 is represented as 00000000 00000010 in binary system.

Negative numbers are represented in 2's complement method.

1's complement of 00000000 00000010 is 11111111 11111101 (Change all 0s to 1 and 1s to 0).

2's complement of 00000000 00000010 is 11111111 11111110 (Add 1 to 1's complement to obtain the 2's complement value).

Therefore, in binary we represent -2 as: 11111111 11111110.

After left shifting it by 2 bits we obtain: 11111111 11111000, and it is equal to "fff8" in hexadecimal system.
View Answer Discuss in Forum Workspace Report

3. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=-3, j=2, k=0, m;
    m = ++i || ++j && ++k;
    printf("%d, %d, %d, %d\n", i, j, k, m);
    return 0;
}
```

A. 2, 2, 0, 1

B. 1, 2, 1, 0

C. -2, 2, 0, 0

D. -2, 2, 0, 1

Answer: Option **D**
Explanation:
Step 1: `int i=-3, j=2, k=0, m;` here variable `i, j, k, m` are declared as an integer type and variable `i, j, k` are initialized to -3, 2, 0 respectively.
Step 2: `m = ++i || ++j && ++k;` here (++j && ++k;) this code will not get executed because ++i has non-zero value.
becomes `m = -2 || ++j && ++k;`
becomes `m = TRUE || ++j && ++k;` Hence this statement becomes TRUE. So it returns '1'(one). Hence m=1.
Step 3: `printf("%d, %d, %d, %d\n", i, j, k, m);` In the previous step the value of variable `'i'` only increemented by '1'(one). The variable `j,k` are not increemented.

Hence the output is "-2, 2, 0, 1".

View Answer Discuss in Forum Workspace Report

4. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x=12, y=7, z;
    z = x!=4 || y == 2;
    printf("z=%d\n", z);
    return 0;
}
```

A. z=0

B. z=1

C. z=4

D. z=2

Answer: Option **B**

Explanation:

Step 1: `int x=12, y=7, z;` here variable `x, y` and `z` are declared as an integer and variable `x` and `y` are initialized to 12, 7 respectively.

Step 2: `z = x!=4 || y == 2;`

becomes `z = 12!=4 || 7 == 2;`

then `z = (condition true) || (condition false);` Hence it returns 1. So the value of `z=1`.

Step 3: `printf("z=%d\n", z);` Hence the output of the program is "z=1".

View Answer Discuss in Forum Workspace Report

5. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    static int a[20];
    int i = 0;
    a[i] = i  ;
    printf("%d, %d, %d\n", a[0], a[1], i);
    return 0;
}
```

A. 1, 0, 1

B. 1, 1, 1

C. 0, 0, 0

D. 0, 1, 0

Answer: Option **C**

Explanation:

Step 1: `static int a[20];` here variable `a` is declared as an integer type and `static`. If a variable is declared as `static` and it will be automatically initialized to value '0'(zero).

Step 2: `int i = 0;` here vaiable `i` is declared as an integer type and initialized to '0'(zero).

Step 3: `a[i] = i ;` becomes `a[0] = 0;`

Step 4: `printf("%d, %d, %d\n", a[0], a[1], i);`

Here a[0] = 0, a[1] = 0(because all staic variables are initialized to '0') and i = 0.

Step 4: Hence the output is "0, 0, 0".

6. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=4, j=-1, k=0, w, x, y, z;
    w = i || j || k;
    x = i && j && k;
    y = i || j &&k;
    z = i && j || k;
    printf("%d, %d, %d, %d\n", w, x, y, z);
    return 0;
}
```

A. 1, 1, 1, 1

B. 1, 1, 0, 1

C. 1, 0, 0, 1

D. 1, 0, 1, 1

Answer: Option **D**

Explanation:

Step 1: `int i=4, j=-1, k=0, w, x, y, z;` here variable `i, j, k, w, x, y, z` are declared as an integer type and the variable `i, j, k` are initialized to 4, -1, 0 respectively.
Step 2: `w = i || j || k;` becomes `w = 4 || -1 || 0;`. Hence it returns TRUE. So, w=1
Step 3: `x = i && j && k;` becomes `x = 4 && -1 && 0;` Hence it returns FALSE. So, x=0
Step 4: `y = i || j &&k;` becomes `y = 4 || -1 && 0;` Hence it returns TRUE. So, y=1
Step 5: `z = i && j || k;` becomes `z = 4 && -1 || 0;` Hence it returns TRUE. So, z=1.
Step 6: `printf("%d, %d, %d, %d\n", w, x, y, z);` Hence the output is "1, 0, 1, 1".
View Answer Discuss in Forum Workspace Report

7. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=-3, j=2, k=0, m;
    m = ++i && ++j || ++k;
    printf("%d, %d, %d, %d\n", i, j, k, m);
    return 0;
}
```

A. 1, 2, 0, 1

B. -3, 2, 0, 1

C. -2, 3, 0, 1

D. 2, 3, 1, 1

Answer: Option **C**

Explanation:

Step 1: `int i=-3, j=2, k=0, m;` here variable `i, j, k, m` are declared as an integer type and variable `i, j, k` are initialized to -3, 2, 0 respectively.
Step 2: `m = ++i && ++j || ++k;`
becomes `m = (-2 && 3) || ++k;`
becomes `m = TRUE || ++k;`.
(++k) is not executed because (-2 && 3) alone return TRUE.
Hence this statement becomes TRUE. So it returns '1'(one). Hence m=1.
Step 3: `printf("%d, %d, %d, %d\n", i, j, k, m);` In the previous step the value of i,j are increemented by '1'(one).

Hence the output is "-2, 3, 0, 1".

View Answer Discuss in Forum Workspace Report

8. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x=4, y, z;
    y = --x;
    z = x--;
    printf("%d, %d, %d\n", x, y, z);
    return 0;
}
```

A. 4, 3, 3

B. 4, 3, 2

C. 3, 3, 2

D. 2, 3, 3

Answer: Option **D**
Explanation:
Step 1: `int x=4, y, z;` here variable `x`, `y`, `z` are declared as an integer type and variable x is initialized to 4.
Step 2: `y = --x;` becomes `y = 3;` because (--x) is pre-decrement operator.
Step 3: `z = x--;` becomes `z = 3;`. In the next step variable `x` becomes 2, because (x--) is post-decrement operator.
Step 4: `printf("%d, %d, %d\n", x, y, z);` Hence it prints "2, 3, 3".
View Answer Discuss in Forum Workspace Report

9. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=3;
    i = i++;
    printf("%d\n", i);
    return 0;
}
```

A. 3

B. 4

C. 5

D. 6

Answer: Option **B**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

10. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int a=100, b=200, c;
    c = (a == 100 || b > 200);
    printf("c=%d\n", c);
    return 0;
}
```

A. c=100

B. c=200

C. c=1

D. c=300

Answer: Option **C**
Explanation:
Step 1: `int a=100, b=200, c;`
Step 2: `c = (a == 100 || b > 200);`
becomes `c = (100 == 100 || 200 > 200);`
becomes `c = (TRUE || FALSE);`
becomes `c = (TRUE);`(ie. c = 1)
Step 3: `printf("c=%d\n", c);` It prints the value of variable `i=1`
Hence the output of the program is '1'(one).

11. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int x=55;
    printf("%d, %d, %d\n", x<=55, x=40, x>=10);
    return 0;
}
```

A. 1, 40, 1

B. 1, 55, 1

C. 1, 55, 0

D. 1, 1, 1

Answer: Option **A**
Explanation:
Step 1: `int x=55;` here variable `x` is declared as an integer type and initialized to '55'.
Step 2: `printf("%d, %d, %d\n", x<=55, x=40, x>=10);`
In printf the execution of expressions is from Right to Left.
here `x>=10` returns TRUE hence it prints '1'.
`x=40` here `x` is assigned to 40 Hence it prints '40'.
`x<=55` returns TRUE. hence it prints '1'.
Step 3: Hence the output is "1, 40, 1".
View Answer Discuss in Forum Workspace Report

12. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=2;
    printf("%d, %d\n", ++i, ++i);
    return 0;
}
```

A. 3, 4

B. 4, 3

C. 4, 4

D. Output may vary from compiler to compiler

Answer: Option **D**
Explanation:

The order of evaluation of arguments passed to a function call is unspecified.

Anyhow, we consider `++i, ++i` are Right-to-Left associativity. The output of the program is 4, 3.

In TurboC, the output will be 4, 3.

In GCC, the output will be 4, 4.

View Answer Discuss in Forum Workspace Report

13. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int k, num=30;
    k = (num>5 ? (num <=10 ? 100 : 200): 500);
    printf("%d\n", num);
    return 0;
}
```

A. 200

B. 30

C. 100

D. 500

Answer: Option **B**

Explanation:

Step 1: `int k, num=30;` here variable `k` and `num` are declared as an integer type and variable `num` is initialized to '30'.

Step 2: `k = (num>5 ? (num <=10 ? 100 : 200): 500);` This statement does not affect the output of the program. Because we are going to print the variable `num` in the next statement. So, we skip this statement.

Step 3: `printf("%d\n", num);` It prints the value of variable `num` '30'

Step 3: Hence the output of the program is '30'

View Answer Discuss in Forum Workspace Report

14. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    char ch;
    ch = 'A';
    printf("The letter is");
    printf("%c", ch >= 'A' && ch <= 'Z' ? ch + 'a' - 'A':ch);
    printf("Now the letter is");
    printf("%c\n", ch >= 'A' && ch <= 'Z' ? ch : ch + 'a' - 'A');
    return 0;
}
```

A. The letter is a
Now the letter is A

B. The letter is A
Now the letter is a

C. Error

D. None of above

Answer: Option **A**

Explanation:

Step 1: `char ch; ch = 'A';` here variable `ch` is declared as an character type an initialized to 'A'.
Step 2: `printf("The letter is");` It prints "The letter is".
Step 3: `printf("%c", ch >= 'A' && ch <= 'Z' ? ch + 'a' - 'A':ch);`

The ASCII value of 'A' is 65 and 'a' is 97.

Here

=> ('A' >= 'A' && 'A' <= 'Z') ? (A + 'a' - 'A'):('A')

=> (TRUE && TRUE) ? (65 + 97 - 65) : ('A')

=> (TRUE) ? (97): ('A')

In printf the format specifier is '%c'. Hence prints 97 as 'a'.

Step 4: `printf("Now the letter is");` It prints "Now the letter is".
Step 5: `printf("%c\n", ch >= 'A' && ch <= 'Z' ? ch : ch + 'a' - 'A');`

Here => ('A' >= 'A' && 'A' <= 'Z') ? ('A') : (A + 'a' - 'A')

=> (TRUE && TRUE) ? ('A') :(65 + 97 - 65)

=> (TRUE) ? ('A') : (97)

It prints 'A'

Hence the output is

The letter is a
Now the letter is A
View Answer Discuss in Forum Workspace Report

15. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=2;
    int j = i + (1, 2, 3, 4, 5);
    printf("%d\n", j);
    return 0;
}
```

A. 4

B. 7

C. 6

D. 5

Answer: Option **B**
Explanation:
Because, comma operator used in the expression `i (1, 2, 3, 4, 5)`. The comma operator has left-right associativity. The left operand is always evaluated first, and the result of evaluation

is discarded before the right operand is evaluated. In this expression 5 is the right most operand, hence after evaluating expression (1, 2, 3, 4, 5) the result is 5, which on adding to i results into 7.