

C programming-2

Task: 1 Find output: [DESCO-SAE(CSE)-2019]

<pre>#include<stdio.h> #define x 9+2/4*3-2*4+(5-4*3) int main () { int i, y; y = 6 + 3 * 3 / 5; i = x * x + y; printf ("%d", i); return 0; }</pre>	Output: -70
--	-------------

Task 2: Unary operator:

```
int main ()
{
    int i = 4;
    printf (" %d\n", i++);
    printf (" %d\n", i++);
    printf (" %d\n", ++i);
    printf (" %d\n", i++);
    printf (" %d\n", i++);
    return 0;
}
```

Example: main ()

```
{
    int i = 4;
    printf (" %d %d %d %d %d", i++, i++, ++i, i++, i++);
    return 0;
}
```

printf ("%d %d %d", a++, a++, ++a); , then which sequence it will print?

We know that printf() uses stack. Means your last argument is treated **first** then 2nd last argument at second place and first argument is treated at last. Like,

++a
a++
a++

Then second thing to remember is **postfix increment/decrement has higher priority than prefix**. This simply means that postfix will assign its value first then increment and prefix will increment first then assign its value just before that statement is completed.

Now let me show you how that statement actually executed.

```
int a = 5;
```

```
printf("%d %d %d", a++, a++, ++a);
```

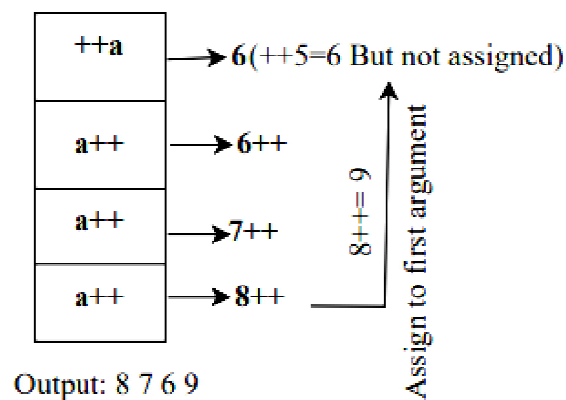
→**Step 1:** last argument is treated first. ++a becomes 6 but that last is not assigned value of 6 yet. It's waiting for whole statement to execute then assign value. Yet we get value of a as 6

→**Step 2:** Now it's time to treat 2nd or I can say 2nd last argument. Told you post increment instantly assigns value then do increment operation. So 2nd argument gets value assigned to 6. Increment is now done and value of a turns out to be 7.

→**Step 3:** That 7 value is assigned to first argument and it turns value of a to 8. Now that all the operations are performed so it's time for pre-increment to do its left out work(assignment of value). So last argument gets value of 8.

So answer of your query is : 7 6 8

Again, `printf("%d %d %d %d", a++, a++, a++, ++a);` // output: 8 7 6 9. See It graphically,



And one more thing to remember: The answer may vary compiler to compiler. Many compiler have different implementations of printf(). So my answer is strict to gcc.

Find output: [JBL-SO(IT)-20]

int i=4; printf("%d %d", ++i, i++); printf("%d", i++);	a) 5,4,6 c) 6,4,6 Ans.: 6, 4, 6	b) 5,7,8 d) 4,5,7
--	---------------------------------------	----------------------

Task 3: Recursion:


Recursion is the process of repeating items in a self-similar way.

```
void recursion() {  
    recursion(); /*function calls itself*/  
}  
  
int main() {  
    recursion();  
}
```

The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Common Mistake : Common Errors in Writing Recursive Methods

The method does not call itself directly or indirectly.

 **Case1:** Non-terminating Recursive Methods (Infinite recursion):

i) No base case.

```
int badFactorial(int x) {  
    return x * badFactorial(x-1);  
}
```

ii) The base case is never reached for some parameter values.

```
int anotherBadFactorial(int x) {  
    if(x == 0)  
        return 1;  
    else  
        return x*(x-1)*anotherBadFactorial(x -2);  
    // When x is odd, we never reach the base case!!  
}
```

Find the output of this program [Ashugonj Power Station Company Limited (APSCL) -2021]

```
#include <stdio.h>  
int recursort (int x) {  
    static int y = 0;  
    if (x <= 0) {  
        return 1;  
    }  
    y = x + y;  
    printf ("%d", y);  
    return recursort(x-2) + recursort(x-3);  
}  
int main () {  
    int result = recursort (5);  
    printf ("%d", result);  
    return 0;}
```