From lecture "Motivation and Introduction to operators in C"

Name of operators	Operators
Other operators	`?:` `&` `*` `sizeof()` `,`

Already completed — Will cover in pointers — Already completed — Will see in a moment

Comma (,) operator

1. Comma operator can be used as a "separator".

Widely used as separator.

Example:

```
int a = 3, b = 4, c = 8;
```

≡

```
int a = 3;
int b = 4;
int c = 8;
```

Multiple definitions in single line.

Comma (,) operator

2 Comma operator can be used as an "operator".

```
int a = (3, 4, 8);

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

Output: 8

Comma operator returns the rightmost operand in the expression and it simply evaluates the rest of the operands and finally reject them.

Comma operator returns the rightmost operand in the expression and it simply evaluates the rest of the operands and finally reject them.

Example:

```
int var = (printf("%s\n", "HELLO!"), 5);

printf("%d", var);
```

This value will be returned to var after evaluating the first operand

It will simply not rejected. First evaluated and then rejected.

```
Output:     HELLO!
            5
```

Comma (,) operator

3. Comma operator is having least precedence among all the operators available in C language.

Example 1:

```
int a;
a = 3, 4, 8;

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

≡

```
int a;
(a = 3), 4, 8;

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

```
Output: 3
```

Comma (,) operator

3 Comma operator is having least precedence among all the operators available in C language.

Example 2:

```
int a = 3, 4, 8;

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

```
Output: error
```

```
int a = 3, 4, 8;
```

is equivalent to

```
int a = 3; int 4; int 8;
```

Error!

Here comma is behaving like a separator.

Comma (,) operator

3 Comma operator is having least precedence among all the operators available in C language.

Example 3:

```
int a;
a = (3, 4, 8);

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

OR

```
int a = (3, 4, 8);

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

Bracket has the highest precedence than any other operator

```
Output: 8
```

Homework problem

What is the output of the following C program fragment?

```c
#include<stdio.h>

int main()
{
    int var;
    int num;

    num = (var = 15, var+35);
    printf("%d", num);
    return 0;
}
```

a) 15
b) 50
c) No output
d) Error