Lecture 30

60 MCQs on

Functions and Storage Classes in ‘C’

What is the default storage class for variables declared inside a function?

a) auto

b) extern

c) static

d) register

auto int x = 20

int x = 20;

auto x = 10; (int)

auto y = 20.5; (double)

Which storage class is used for variables that need to retain their values across multiple function calls?

a) register

b) extern

c) static ← Data Segment

d) auto

Which storage class limits the scope of a variable to the current file?

a) extern

b) static

c) register

d) auto

main.c → extern int x;
extern int y;

a.c → int x = 10;

b.c → int y = 20

Static Variable can not shared with other files

Which storage class hints to the compiler to store the variable in CPU registers for faster access?

a) register

b) auto

c) extern

d) static

register int x = 10;

or

register y = 20;

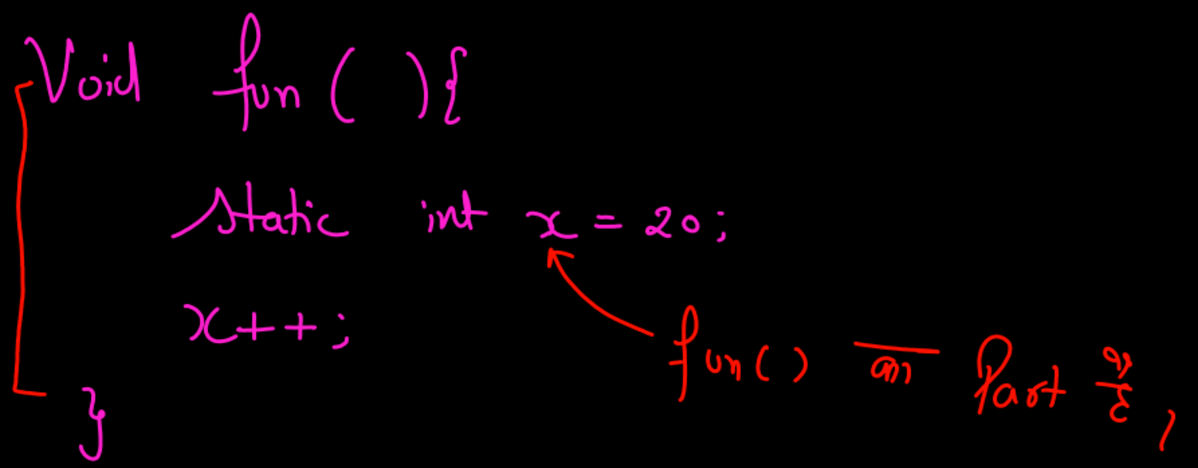
What is the linkage of a variable declared with the static keyword outside any function?

a) External linkage

b) Internal linkage

c) No linkage

d) Local linkage



If a static variable is declared inside a function, what will its initial value be if it's not explicitly initialized?

a) Undefined

b) Random garbage value

c) 0

d) Depends on the compiler

What is true about a register variable in C?

a) It is guaranteed to be stored in a CPU register

b) It can be accessed using the & operator

c) It can be declared globally

d) The compiler may ignore the register request

Can not used with register variable

```
register char x= 'A' ;
printf ("%p", &x) ;
```

Error

Which storage class would you use if you want a variable to be accessible across different source files but defined only once?

a) auto

b) static

c) register

d) extern

Which of the following statements is correct about an auto variable?

a) It is automatically initialized to 0

b) It has global scope

c) It is automatically destroyed when the function ends

d) It can only be used in global declarations

local Scope

Which statement about a static variable in a function is true?

a) It is allocated on the stack x ← data segment

b) Its value persists between function calls

c) It is initialized each time the function is called

d) It has external linkage

⇓

Static Variable does not have external linkage.

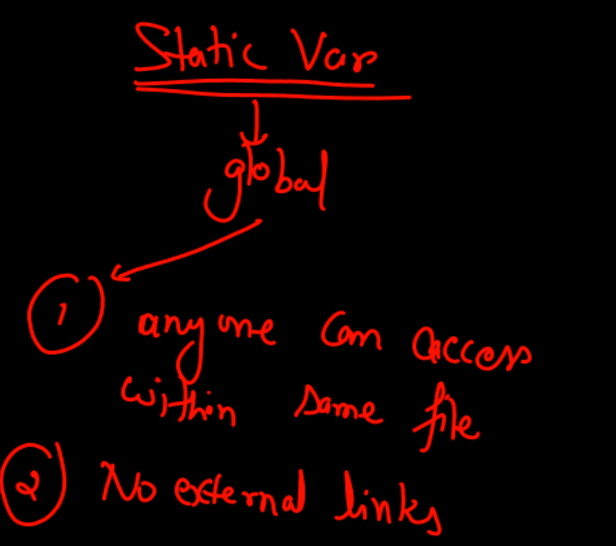
If a global variable is declared as static, what is its scope?

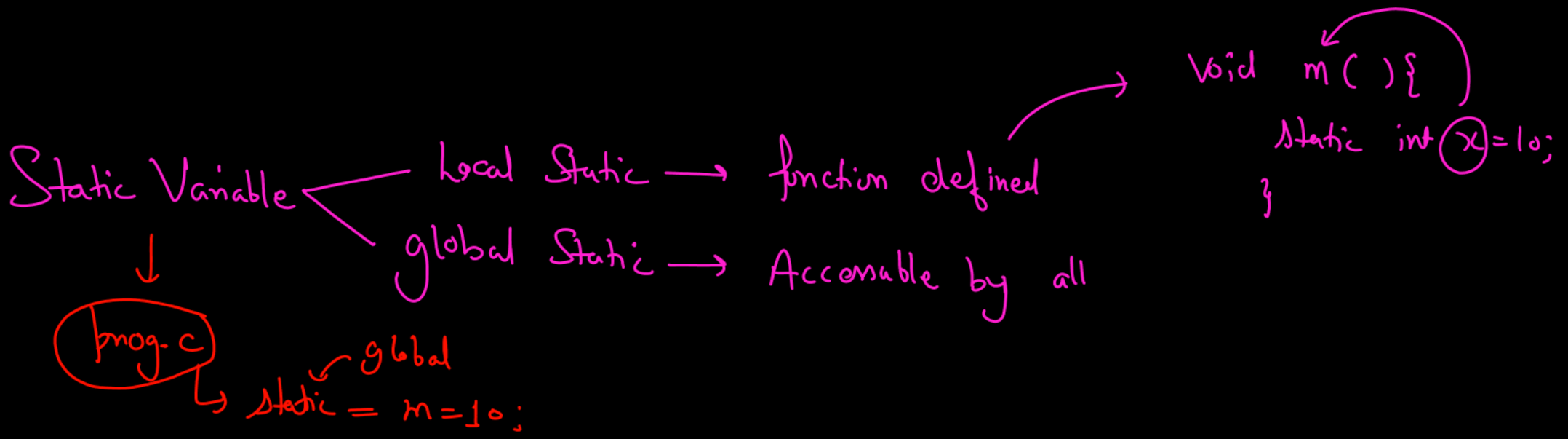
a) It is accessible from any source file in the program

b) It is only accessible within the file it is declared in

c) It is accessible within the block where it is declared

d) It has no linkage





"main.c"

```
Void fun( ){
    Static int x = 0;
    printf("%d", x);
    x++;
}
Static int y = 100;
int main( ){
    fun( );
    printf("%d", y);
    y++;
    return 0;
}
```

local Static

local Static Variable

↓

x ⇒ local to the function fun()

only fun() can access the value of x.

global Static

↳ Any function can access the variable

Output

0
100

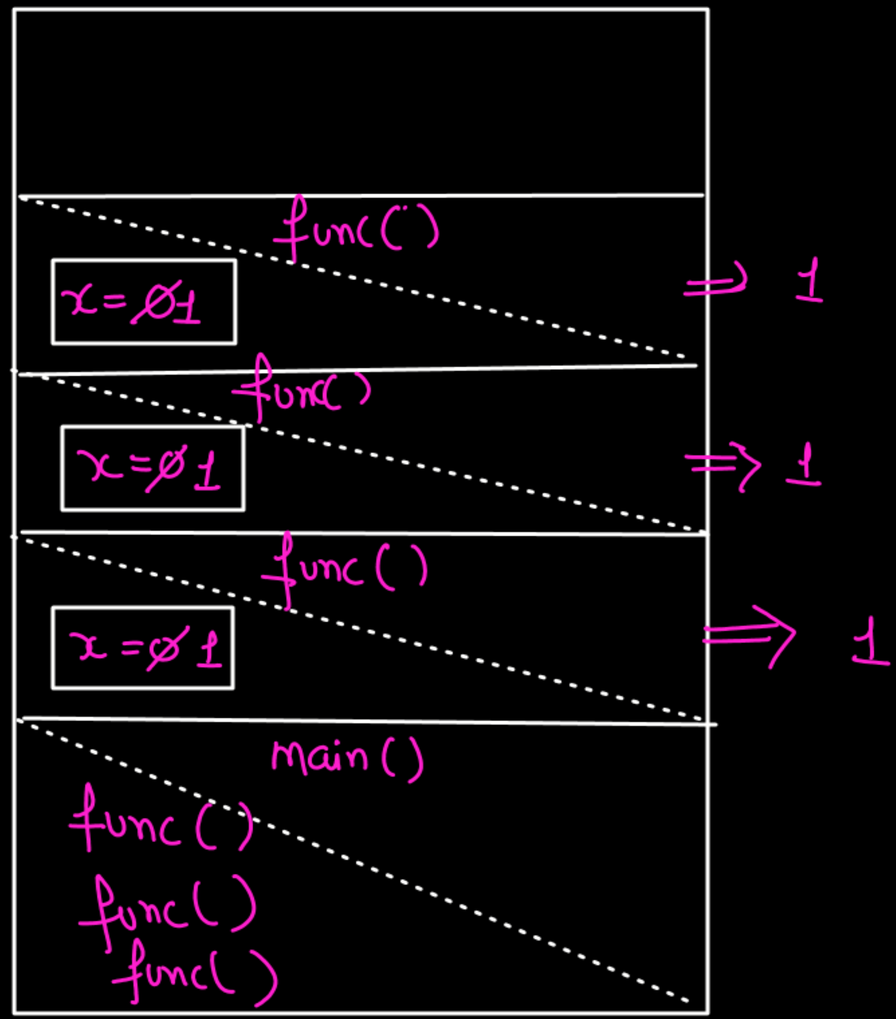
x = 1
y = 101

What will be the output of the following code?

```
#include <stdio.h>
void func() {
    auto int x = 0;
    x++;
    printf("%d ", x);
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

entry point

a) 1 2 3
b) 1 1 1
c) 0 0 0
d) Compilation error

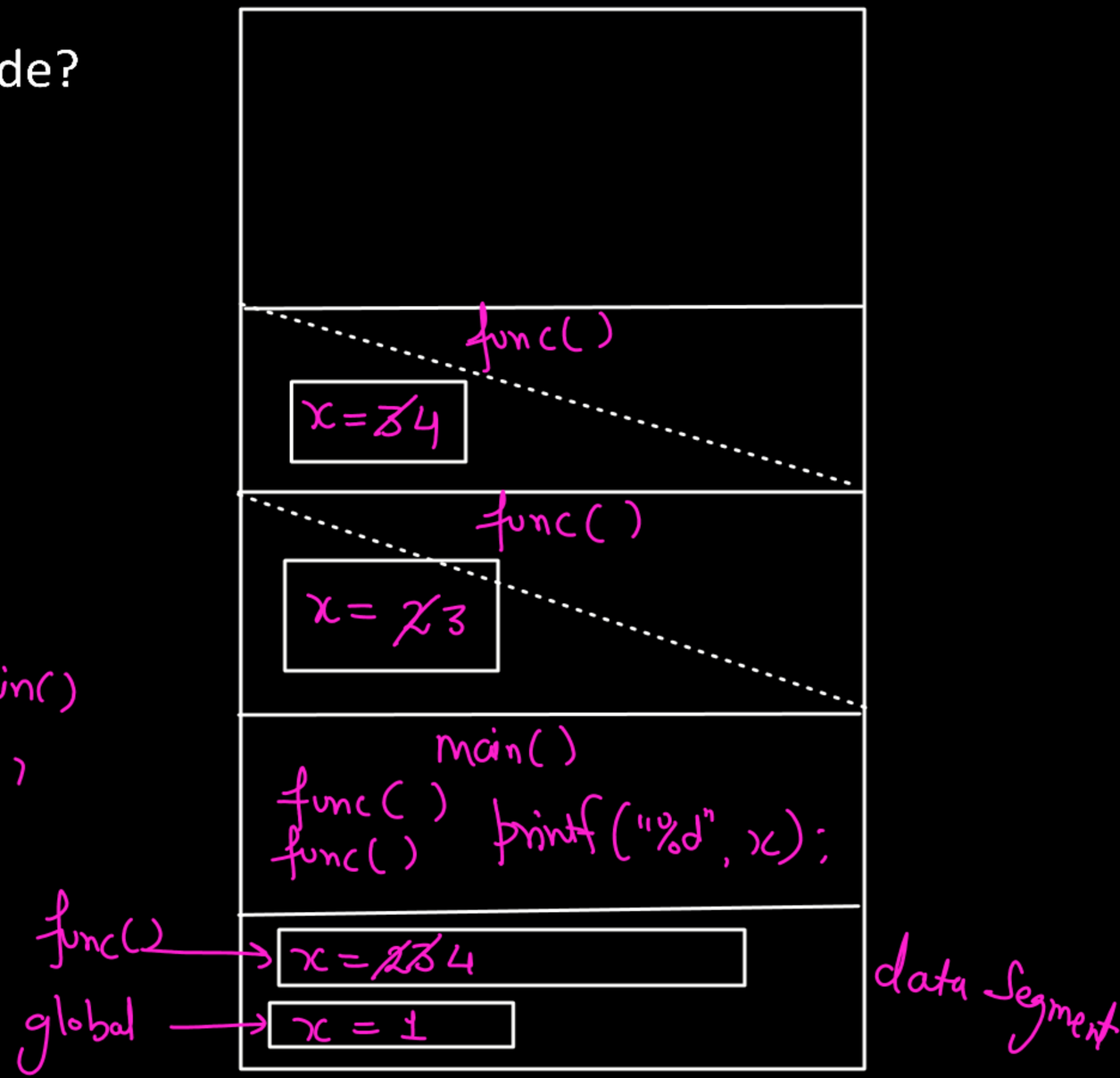


What will be the output of the following code?

```
#include <stdio.h>
int x = 1;
void func() {
    static int x = 2;
    printf("%d ", x);
    x++;
}
int main() {
    func();
    func();
    printf("%d", x);
    return 0;
}
```

a) 2 3 3
b) 2 2 1
c) 2 3 1
d) 3 3 1

output
2 3 1



What will be the output of the following code?

```
#include <stdio.h>
int x = 10;
int main() {
    extern int x;
    {
        int x = 20;
        printf("%d ", x);
    }
    printf("%d", x);
    return 0;
}
```

a) 10 20
b) 20 10
c) 20 20
d) 10 10

entry point

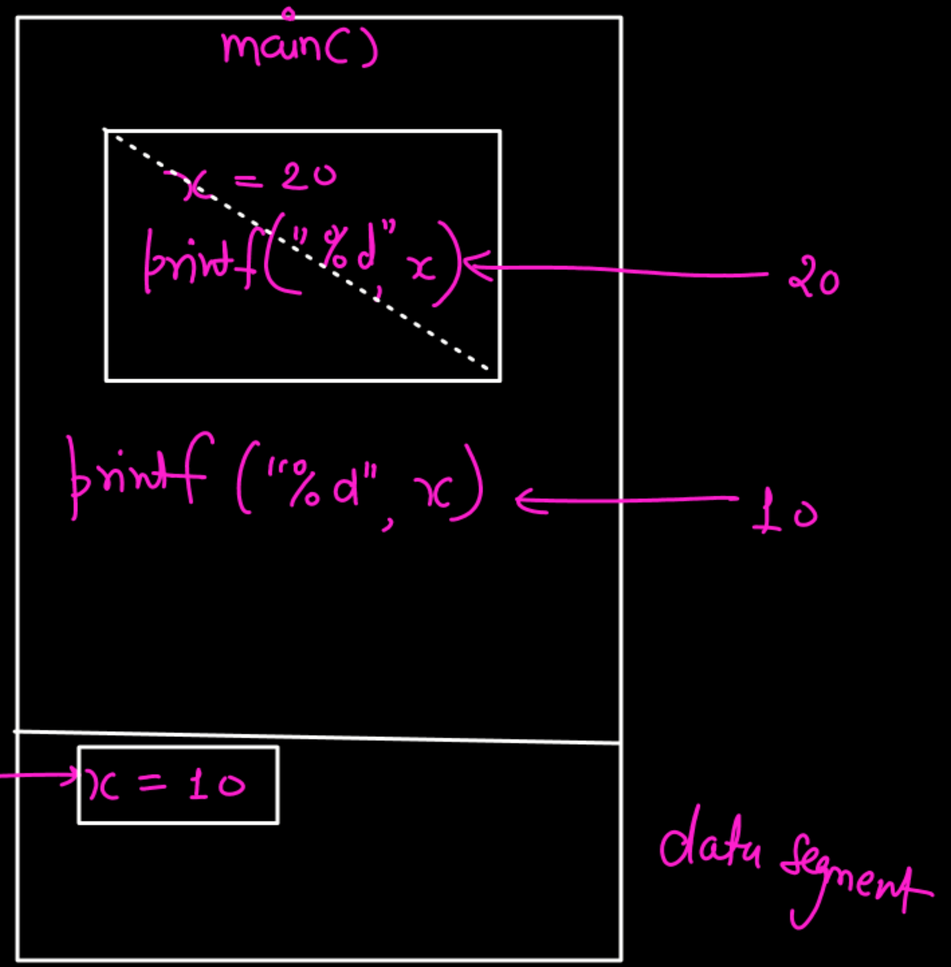
Outside

10

destroyed when block executed.

10

* If there is no external variable or file linkage then the global variable is called.



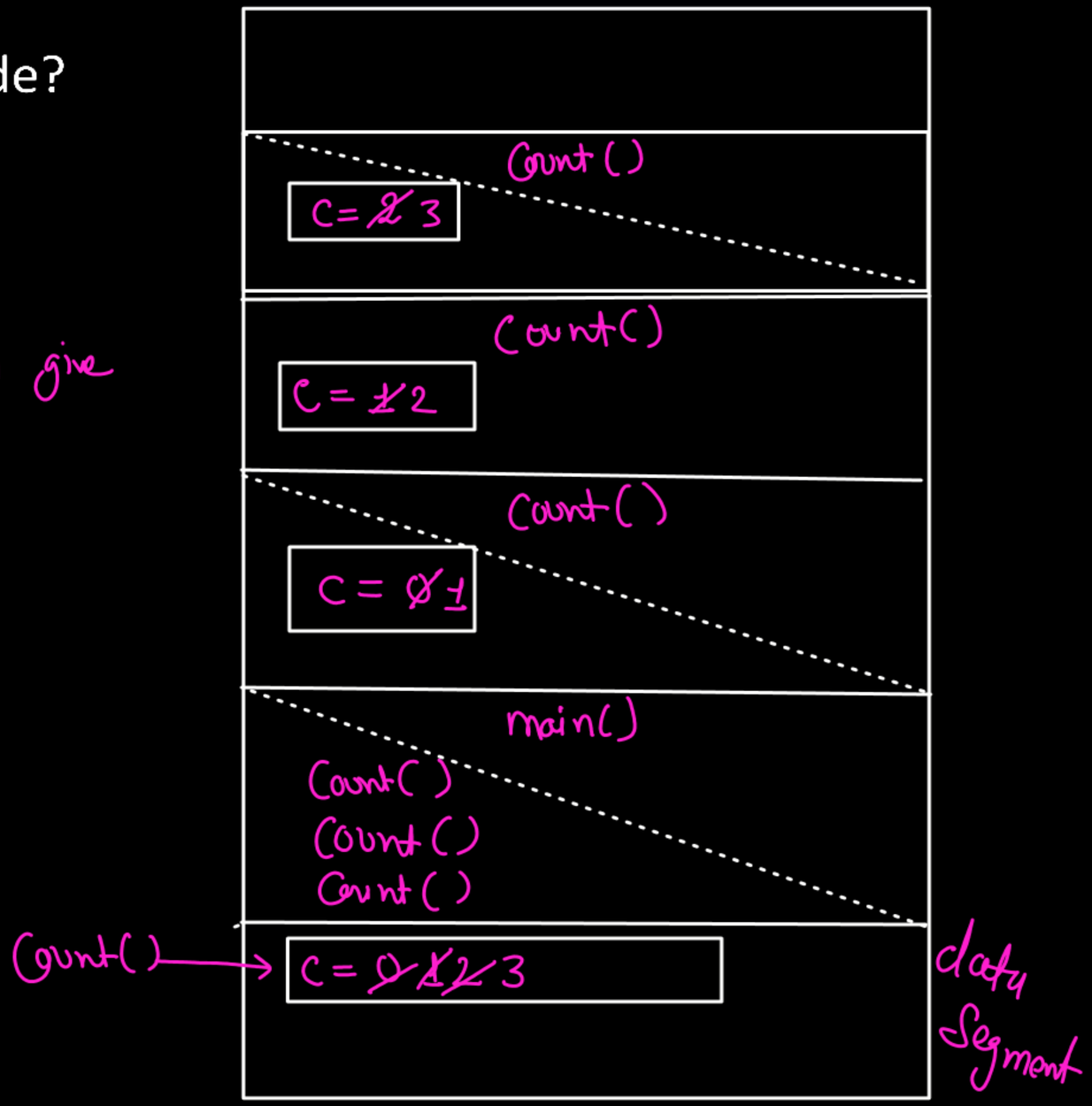
What will be the output of the following code?

```
#include <stdio.h>
int count() {
    static int c = 0;
    return ++c;
}
int main() {
    printf("%d ", count());
    printf("%d ", count());
    printf("%d", count());
    return 0;
}
```

increment first then give the value

1 ⇒ 1
2 ⇒ 2
3 ⇒ 3

a) 1 1 1
b) 1 2 3
c) 0 1 2
d) 2 3 4



Which of the following scenarios would cause a compiler error in C?

a) Declaring a register variable globally

b) Declaring a static variable globally

c) Using extern to reference a variable in another file

d) Declaring a static variable inside a function

Register Variables are local (always)
↳ They can only defined inside a function.

What is true about the following code?

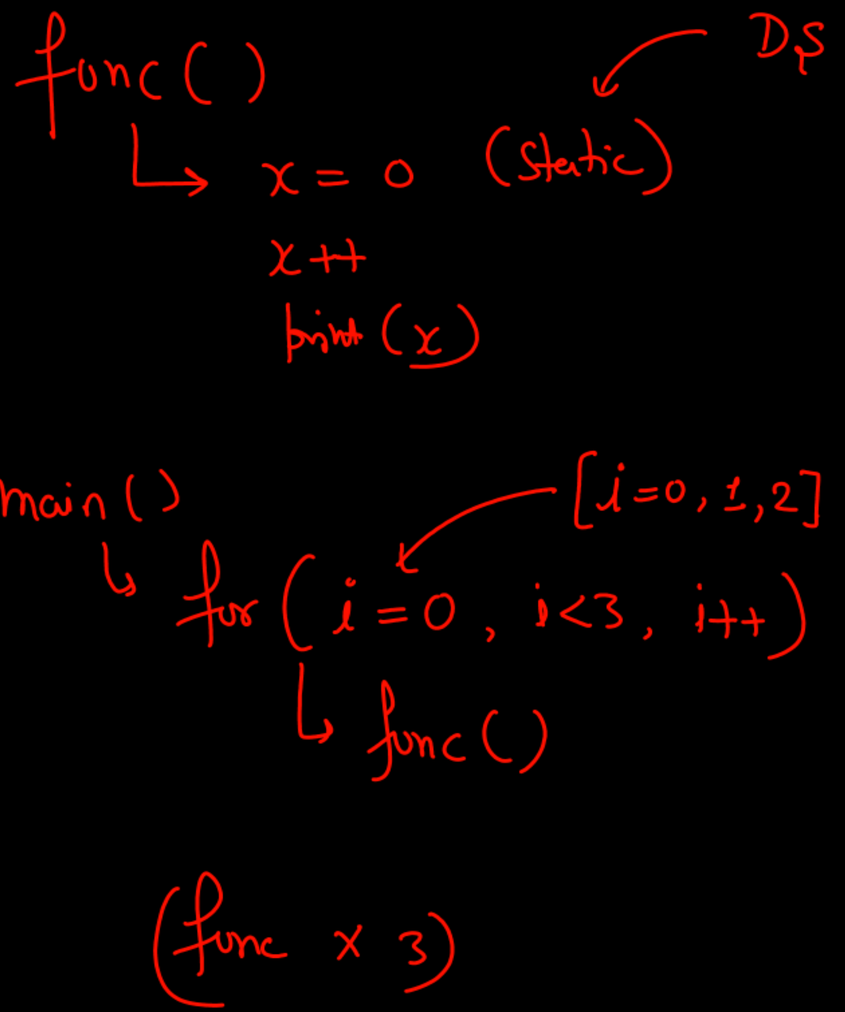
```
#include <stdio.h>
void func() {
    static int x = 0;
    x++;
    printf("%d ", x);
}
int main() {
    for (int i = 0; i < 3; i++) {
        func();
    }
    return 0;
}
```

a) x is reinitialized to 0 each time func is called ✗

b) x is initialized only once and increments on each call

c) x is stored in the register memory

d) x is inaccessible outside main()



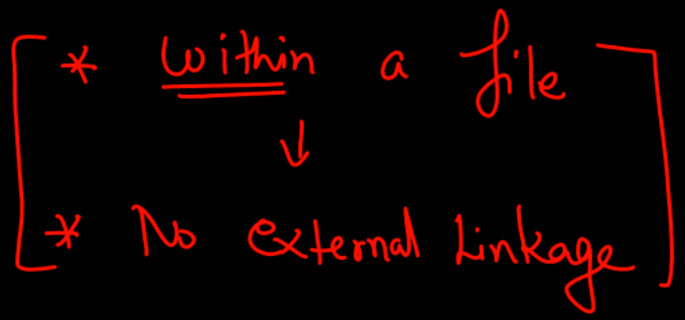
Which of the following would be the scope and linkage of a static variable declared at file level in C?

a) File scope, internal linkage

b) Block scope, no linkage

c) Global scope, external linkage

d) Function scope, external linkage



In which of the following cases will the compiler most likely ignore the register keyword request?

a) When the variable is declared in a loop

b) When the variable is of type int

c) When the variable is accessed frequently

d) When the variable has a large data type, such as an array

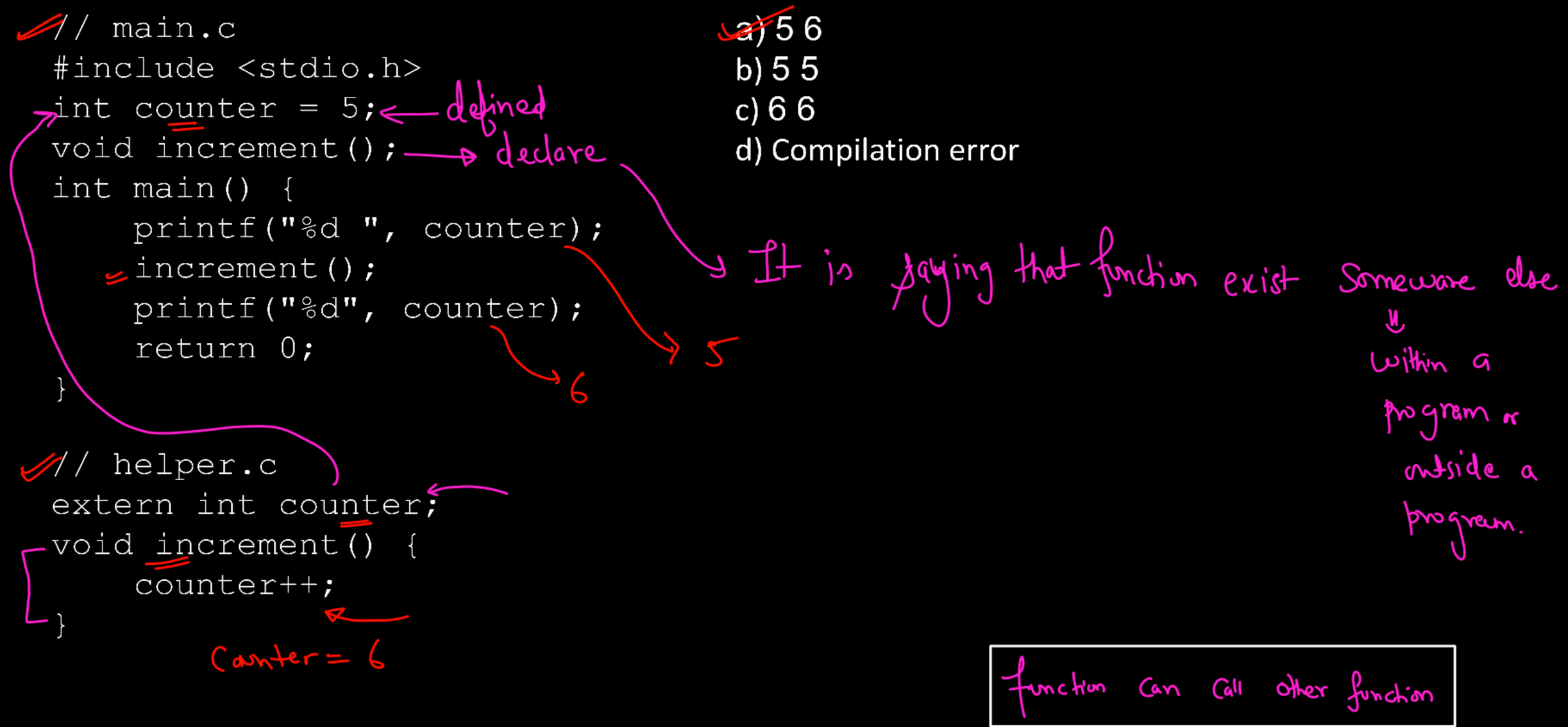
Register → CPU ← less

What is the result of the following program if the initial global variable counter is defined in main.c but accessed in helper.c with extern?

```c
// main.c
#include <stdio.h>
int counter = 5;
void increment();
int main() {
    printf("%d ", counter);
    increment();
    printf("%d", counter);
    return 0;
}
```

```c
// helper.c
extern int counter;
void increment() {
    counter++;
}
```

a) 5 6
b) 5 5
c) 6 6
d) Compilation error



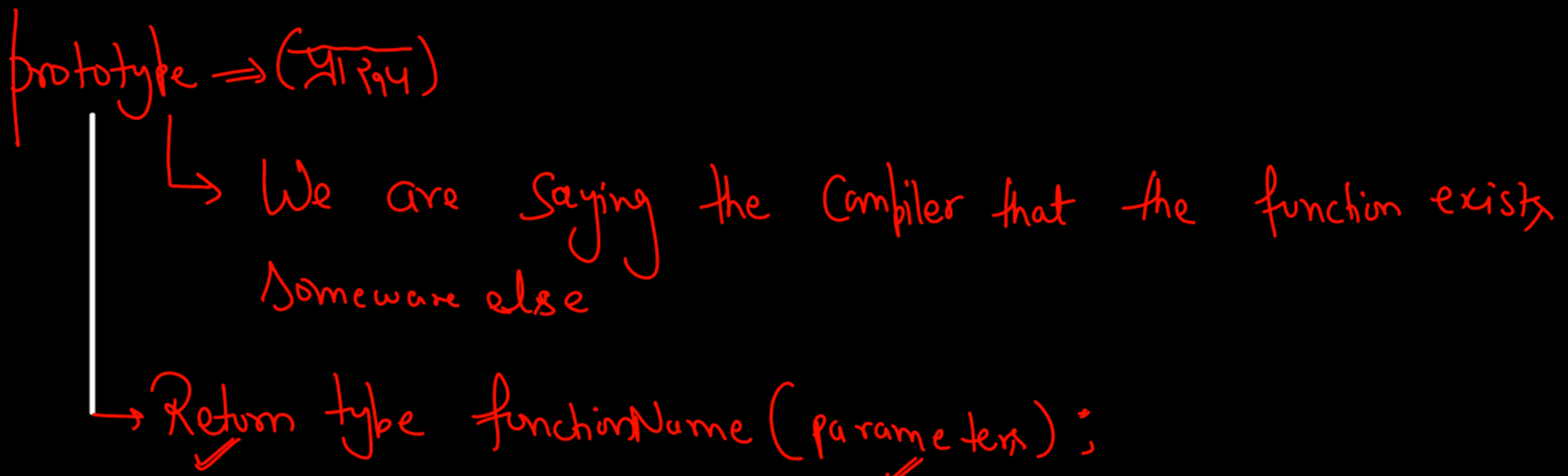
Which of the following is true about function prototypes in C?

a) Function prototypes must specify parameter names.

b) Function prototypes must be defined after the main function. ✗

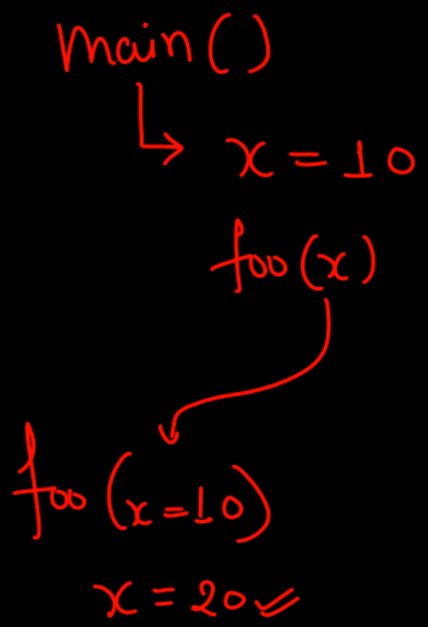
c) A function prototype provides the compiler with the function's return type and parameter types. ✓

d) Function prototypes are optional if the function is declared within main(). ✗



What is the output of the following code?

```
#include <stdio.h>
void foo(int x) {
    x = 20;
}
int main() {
    int x = 10;
    foo(x);
    printf("%d", x);
    return 0;
}
```



a) 10

b) 20

c) Compilation error

d) Undefined behavior

Which of the following statements about recursive functions in C is correct?

a) Recursive functions can only call other functions, not themselves. ✗

b) Recursive functions must have a base case to avoid infinite recursion. ✓

c) Recursive functions in C are faster than loops for all operations.

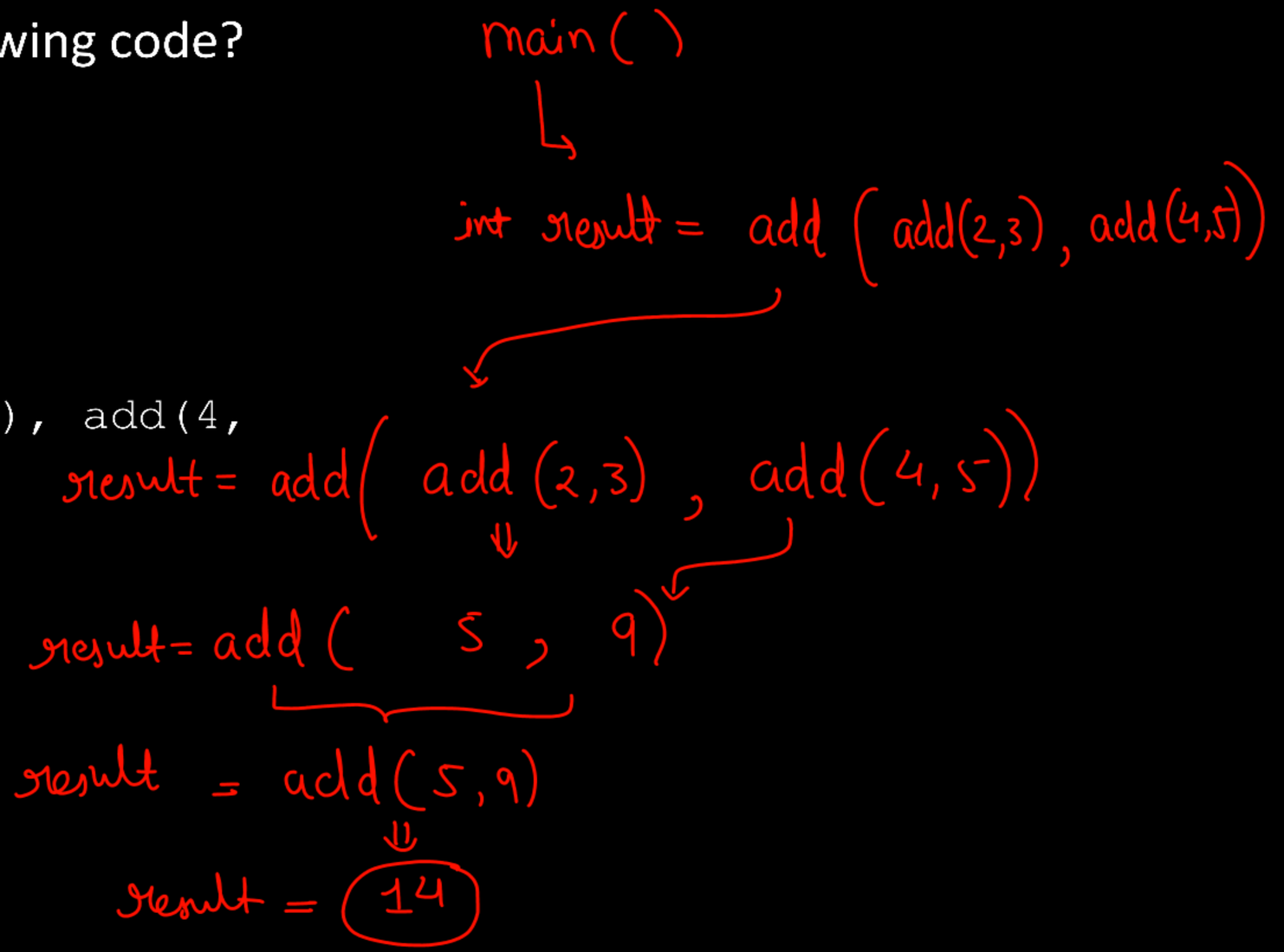
d) Recursive functions do not use stack memory.

What will be the output of the following code?

```
#include <stdio.h>
int add(int x, int y) {
    return x + y;
}
int main() {
    int result = add(add(2, 3), add(4,
5));
    printf("%d", result);
    return 0;
}
```

14

a) 9
b) 14
c) 19
d) Compilation error



Which of the following statements about inline functions in C is correct?

No Activation Record

a) Inline functions are always compiled as inline, reducing function call overhead.

b) Inline functions are ignored by the compiler if they contain loops or recursion.

c) Inline functions require a different syntax than regular functions.

d) Inline functions are stored in a different memory section than regular functions.

⇒ Inline functions = The function that are used to perform specific task without making seperate activation records in the memory.

```
inline int add (int x, int y){
    return x+y;
}
```

x=10, y=20

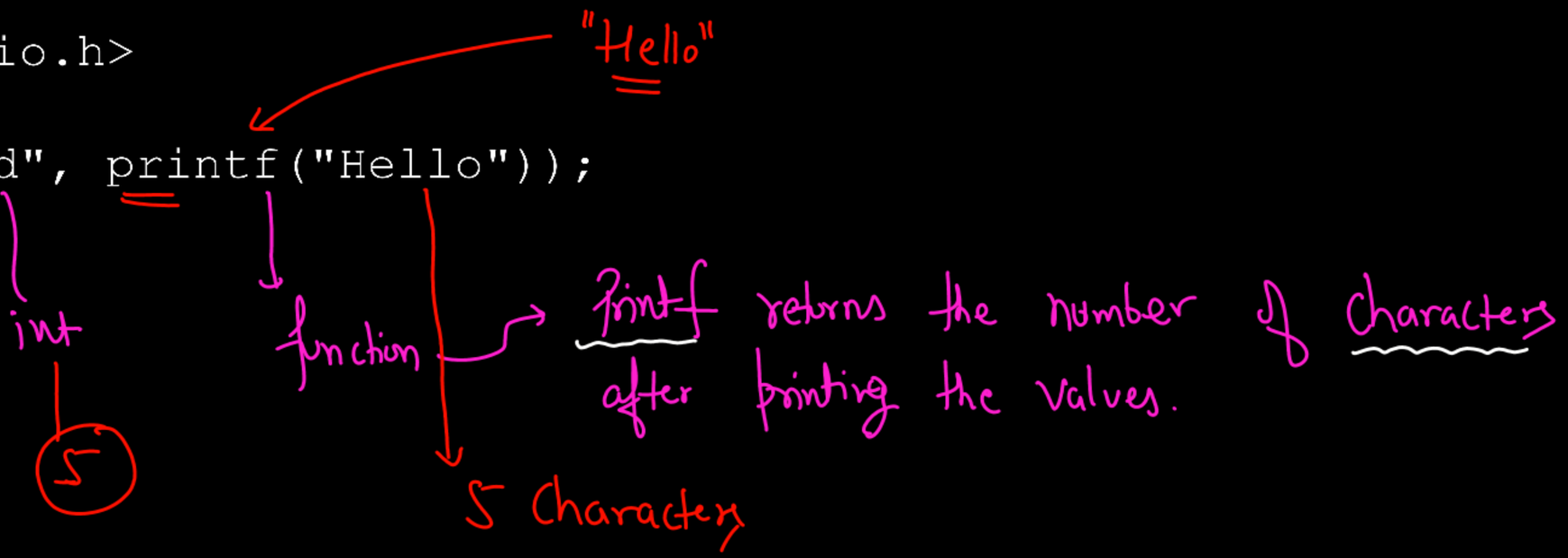
```
main(){
    printf("%d", add(10, 20));
    return 0;
}
```

(x+y) x=10, y=20

Replace with the return expression

What will be the output of the following code?

```
#include <stdio.h>
int main() {
    printf("%d", printf("Hello"));
    return 0;
}
```



a) Hello5

b) Hello6

c) Hello0

d) 5Hello

What will be the output of the following code?

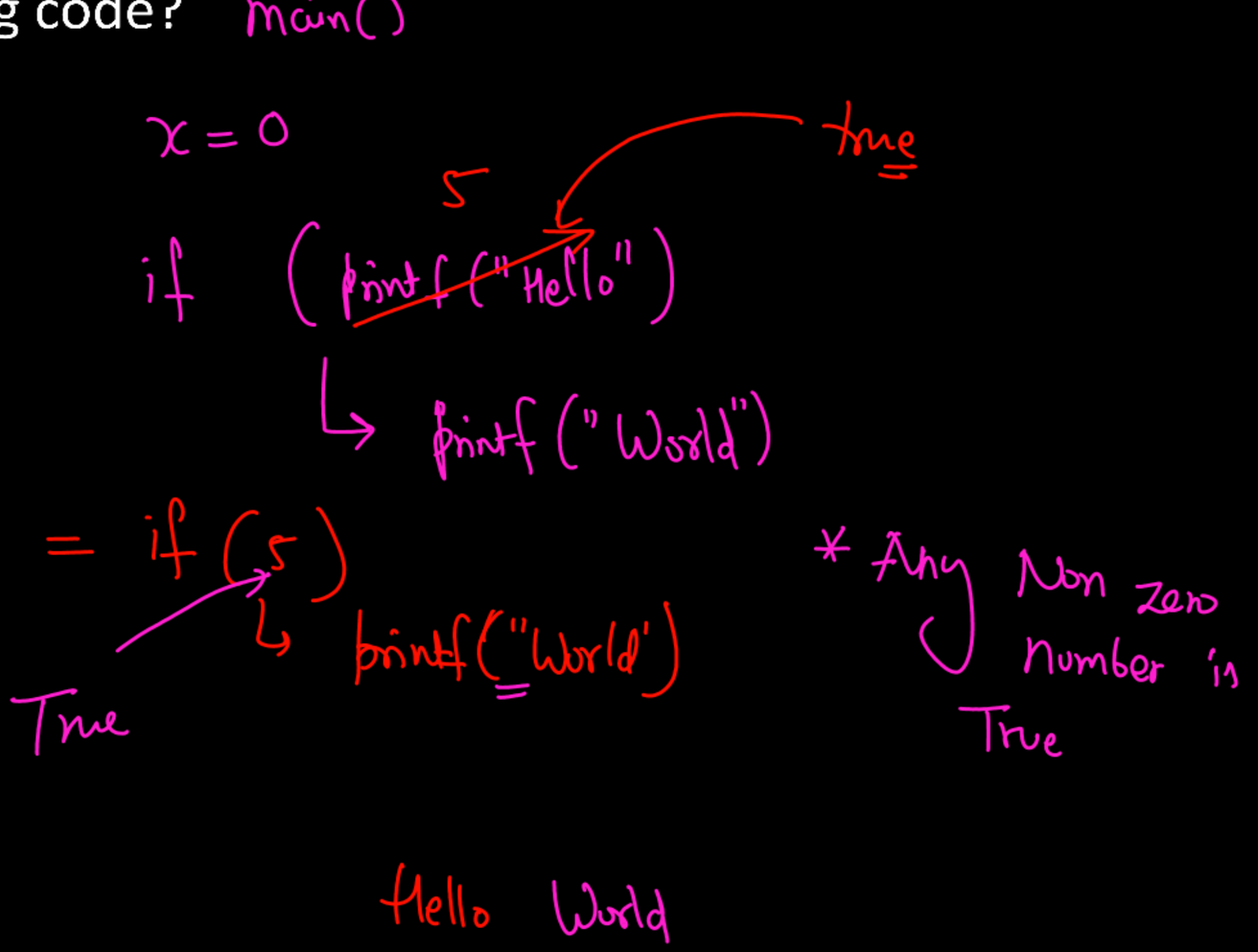
```c
#include <stdio.h>
int main() {
    int x = 0;
    if (printf("Hello")) {
        printf("_World");
    }
    return 0;
}
```

a) Hello World

b) Hello

c) World

d) No output



Which of the following best describes a function's scope in C?

a) The scope of a function is always global.

b) The scope of a function is limited to the file it's declared in.

c) The scope of a function depends on its return type.

d) The scope of a function depends on the parameters it receives.

functions are global, having global scope, can be accessed from another file linked with a C program.

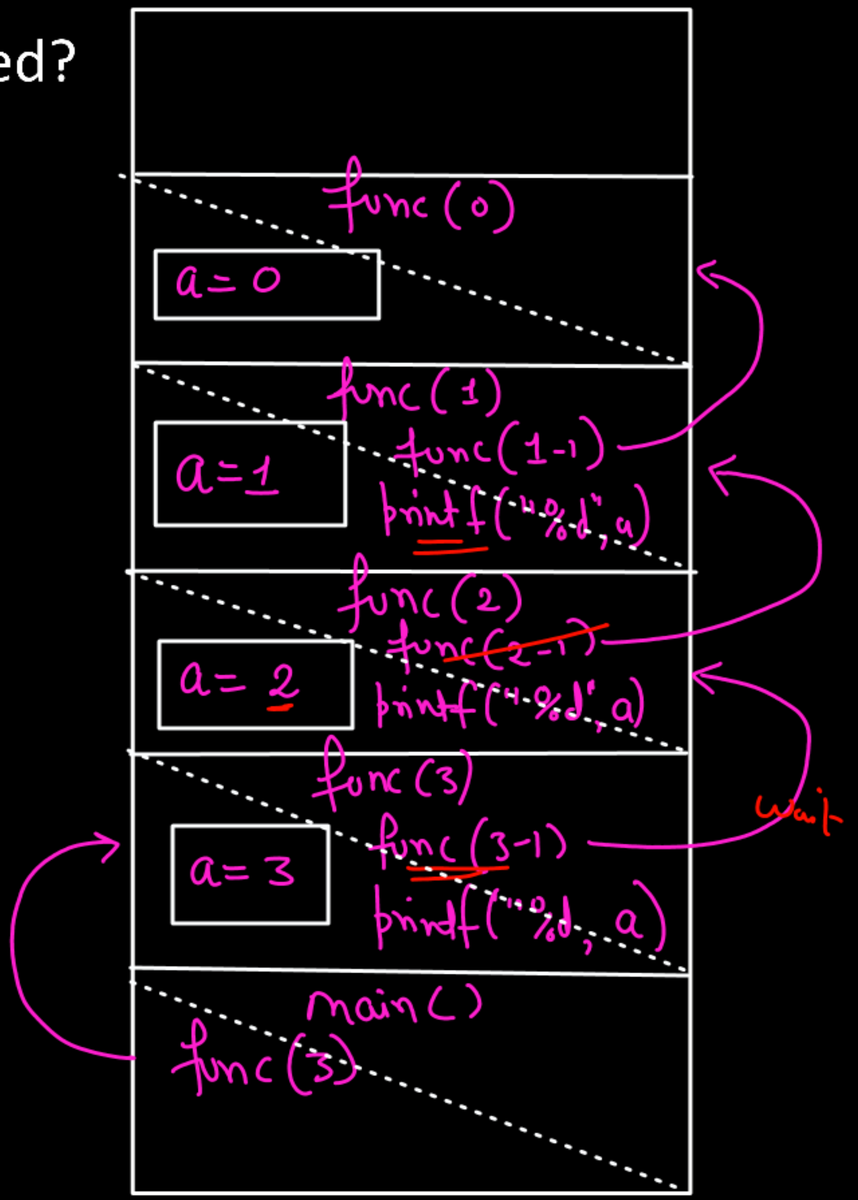
⇒ "Except Static function"

What will be the output of the following code if executed?

```
#include <stdio.h>
void func(int a) {
    if (a > 0) {
        printf("%d ", a);
        func(a - 1);
    }
    printf("%d ", a);
}
int main() {
    func(3);
    return 0;
}
```

a) 3 2 1 0
b) 3 2 1 0 1 2 3
c) 3 2 1
d) 3 3 2 2 1 1

output
3 2 1 0 1 2 3



What is the result of the following code?

```
#include <stdio.h>

void display(int a) {
    printf("%d ", a);
}

void display(float b) {
    printf("%f ", b);
}

int main() {
    display(5);
    display(5.0);
    return 0;
}
```

In C program, it is not possible to define two function having same name.

a) 5 5.000000
b) 5 5
c) Compilation error
d) 5 5.000000 5

What will be the output of the following code?

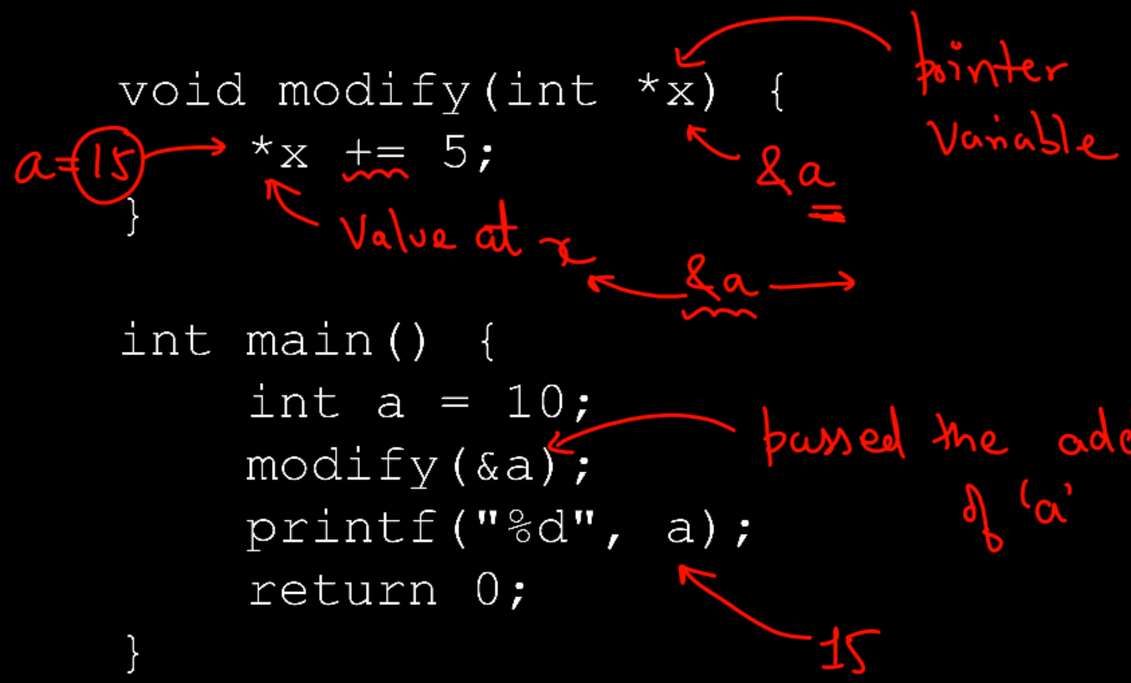
```
#include <stdio.h>

void modify(int *x) {
    *x += 5;
}

int main() {
    int a = 10;
    modify(&a);
    printf("%d", a);
    return 0;
}
```

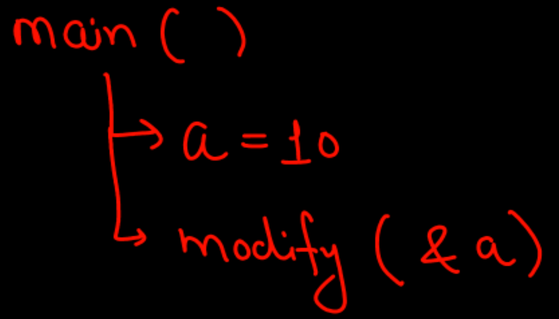
a) 5

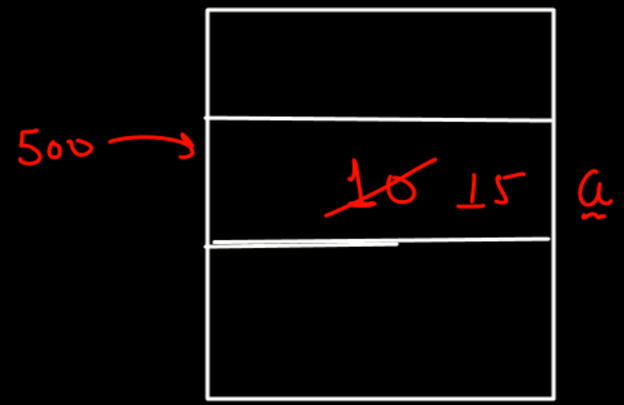
b) 10

c) 15

d) 0







Which of the following is true about local variables in C functions?

a) They retain their values between function calls. X

b) They are initialized by default to zero. X

c) They are accessible from any function within the file. X

d) Their lifetime is limited to the function in which they are defined. ✓

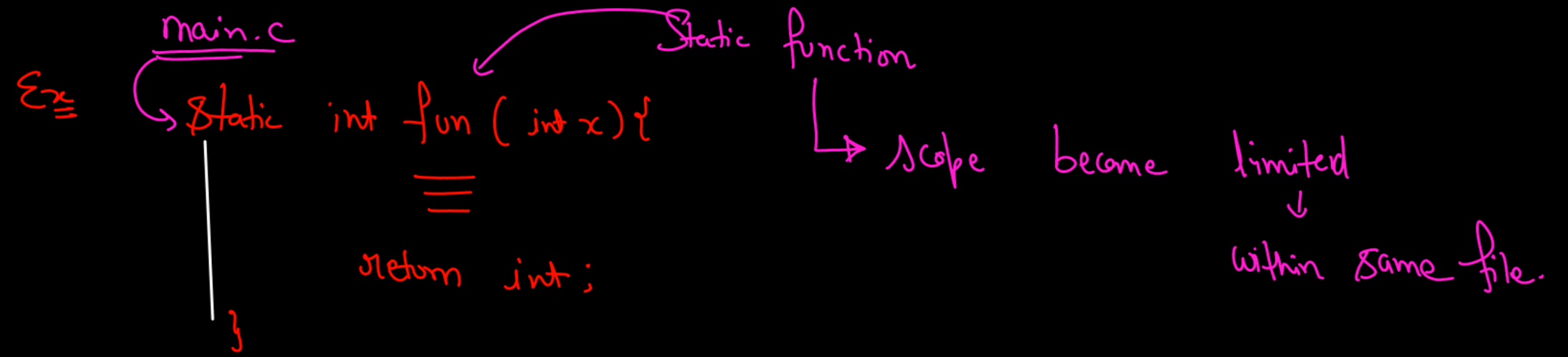
What is the effect of declaring a function as static in C?

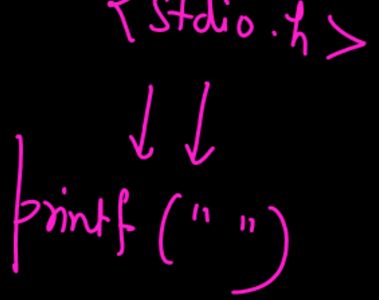
a) It can be called from any file. ✗

b) It can only be called within the same source file. ✓

c) It is stored in static memory.

d) It has a global scope.





What will be the output of the following code?

```
#include <stdio.h>
void fun(int x) {
    if (x > 0) {
        fun(x - 1);
    }
    printf("%d ", x);
}
int main() {
    fun(3);
    return 0;
}
```

H.W

Do Yourself

a) 0 1 2 3

b) 3 2 1 0

c) 0 1 2 3 4

d) 3 2 1 0 0

What will happen if a function in C does not return a value but has a return type other than void?

a) The program will compile without any warnings.

b) The program will result in undefined behavior.

c) The program will return a default value of zero.

d) The program will throw a compilation error.

```
void fun() {

    return x;
}
```

What will be the output of the following code?

```
#include <stdio.h>
void f(int a, int b) {
    a += b;
    b += a;
    printf("%d %d\n", a, b);
}
int main() {
    int x = 5, y = 10;
    f(x, y);
    printf("%d %d\n", x, y);
    return 0;
}
```

a) 15 25 then 5 10
b) 15 15 then 5 10
c) 15 25 then 15 25
d) 5 10 then 5 10

What will happen if a function is called recursively too many times without a base case?

a) The program will run successfully.

b) A stack overflow error will occur.

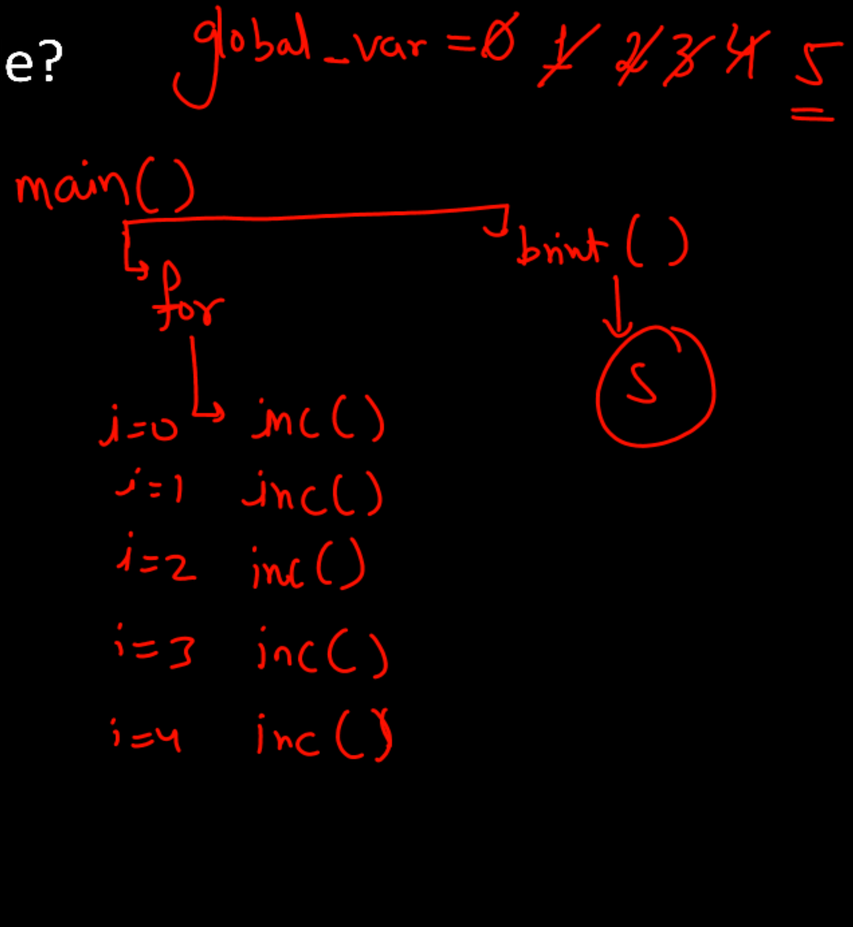
c) The program will hang indefinitely.

d) The function will terminate successfully.

What will be the output of the following code?

```c
#include <stdio.h>
int global_var = 0;
void increment() {
    global_var++;
}
int main() {
    for (int i = 0; i < 5; i++) {
        increment();
    }
    printf("%d", global_var);
    return 0;
}
```

- a) 5
- b) 0
- c) 1
- d) 10



Which of the following statements is true regarding the scope of variables in C functions?

a) Global variables can only be accessed in the function where they are defined.

b) Local variables are accessible throughout the entire program.

c) Variables defined in a function can be accessed by other functions if declared as extern.

d) Static local variables are destroyed when the function exits.

What will be the output of the following code?

```
#include <stdio.h>
int counter() {
    static int count = 0;
    count++;
    return count;
}
int main() {
    printf("%d ", counter());
    printf("%d ", counter());
    printf("%d ", counter());
    return 0;
}
```

Repeated

a) 1 2 3

b) 1 1 1

c) 0 1 2

d) 1 2 2

What will be the output of the following code?

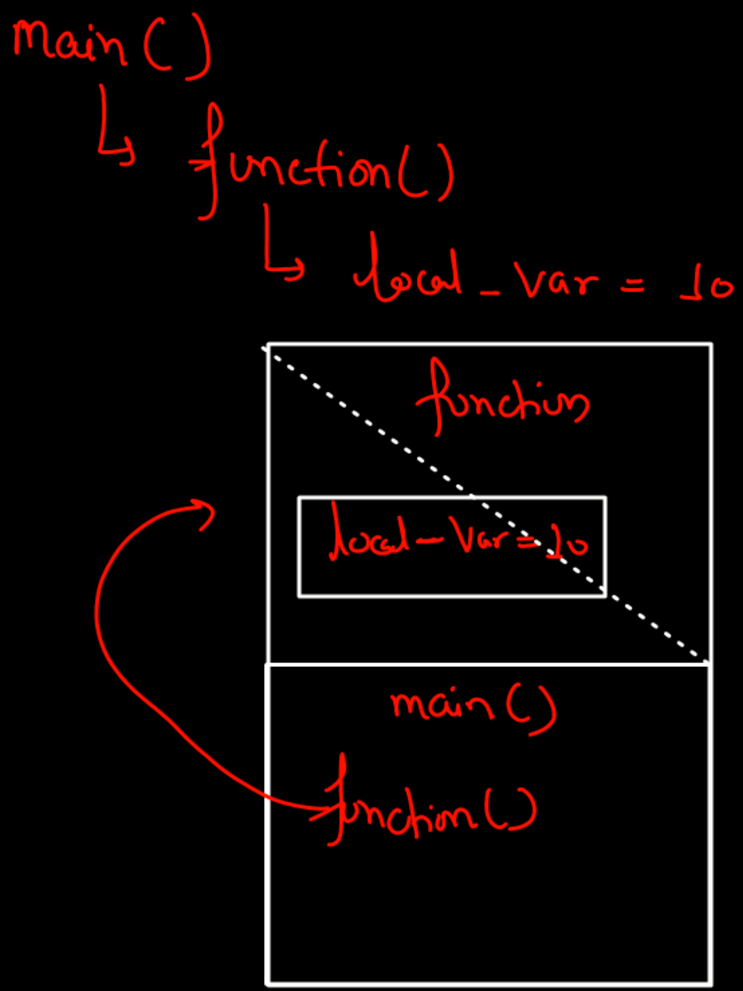
```
#include <stdio.h>
void function() {
    int local_var = 10;
    printf("%d ", local_var);
}
int main() {
    function();
    printf("%d ", local_var);
    return 0;
}
```

a) 10 10

b) 10 0

c) 10 undefined

d) Compilation error



What is the primary purpose of the register **storage class in C?**

a) To reserve memory in the heap.

b) To store variables that are frequently accessed.

c) To keep variables persistent throughout the program.

d) To manage memory manually.

register int m = 10;

(100,000) access

What will be the output of the following code?

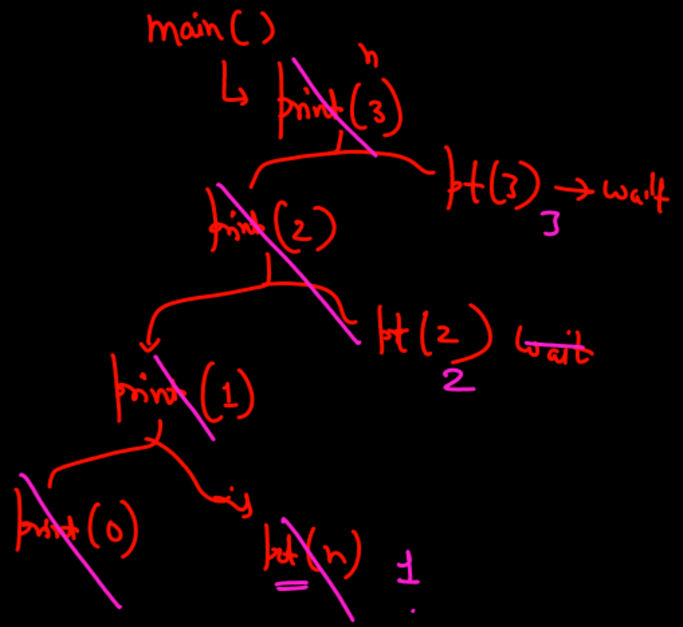
```
#include <stdio.h>
void print(int n) {
    if (n > 0) {
        print(n - 1);
        printf("%d ", n);
    }
}
int main() {
    print(3);
    return 0;
}
```

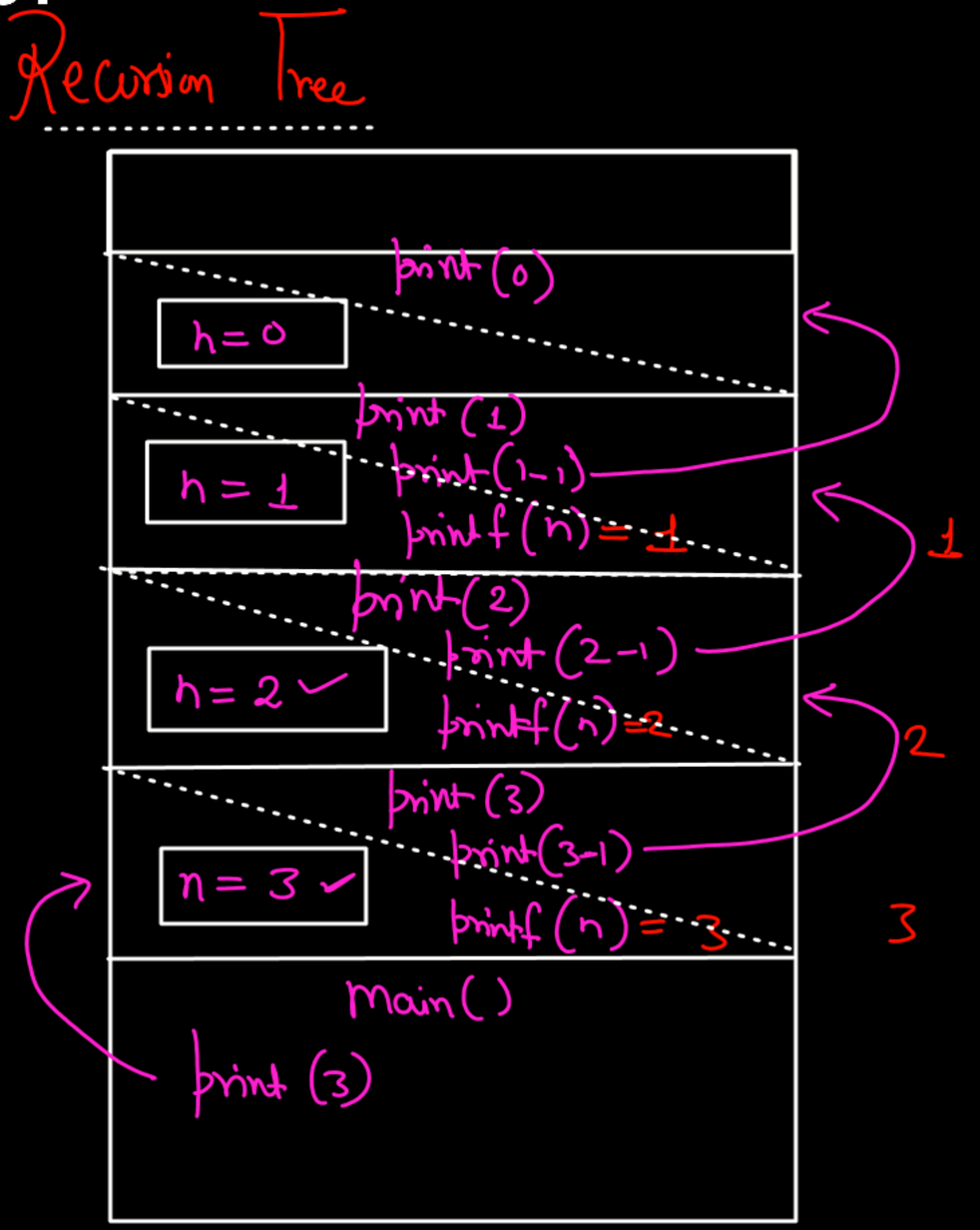
a) 0 1 2 3

b) 1 2 3

c) 3 2 1

d) 0 1 2





Which of the following best describes recursion in functions? (Base Case, Recursive Case)

a) A function calls itself with no base case. ✗

b) A function that calls another function. ✗

c) A function that does not return a value. ✗

d) A function that calls itself with a base case. ✓

What is the output of the following code?

```
#include <stdio.h>
int f(int a, int b) {
    if (b == 0)
        return a;
    return f(b, a % b);
}
int main() {
    printf("%d\n", f(18, 24));
    return 0;
}
```

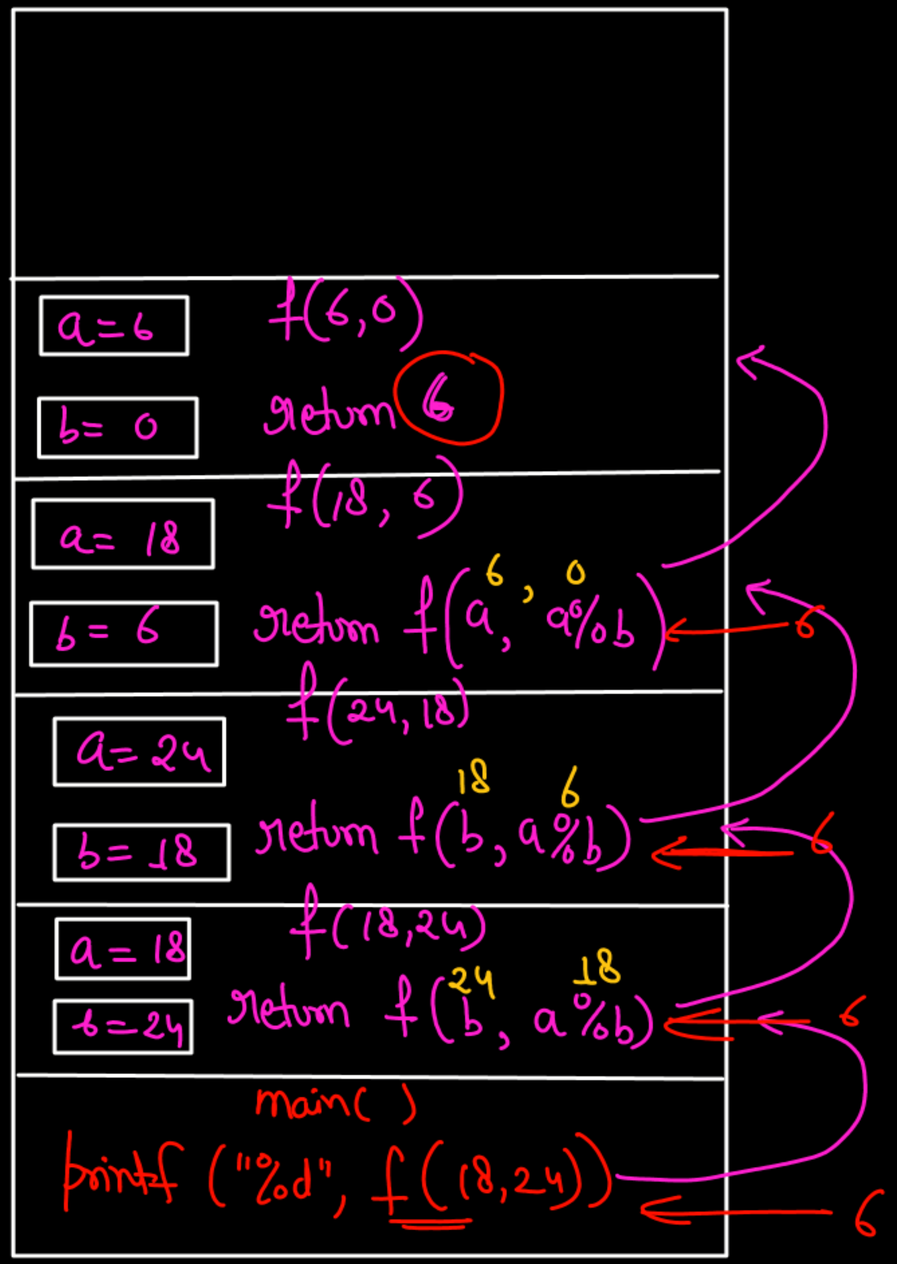
6

A) 6

B) 18

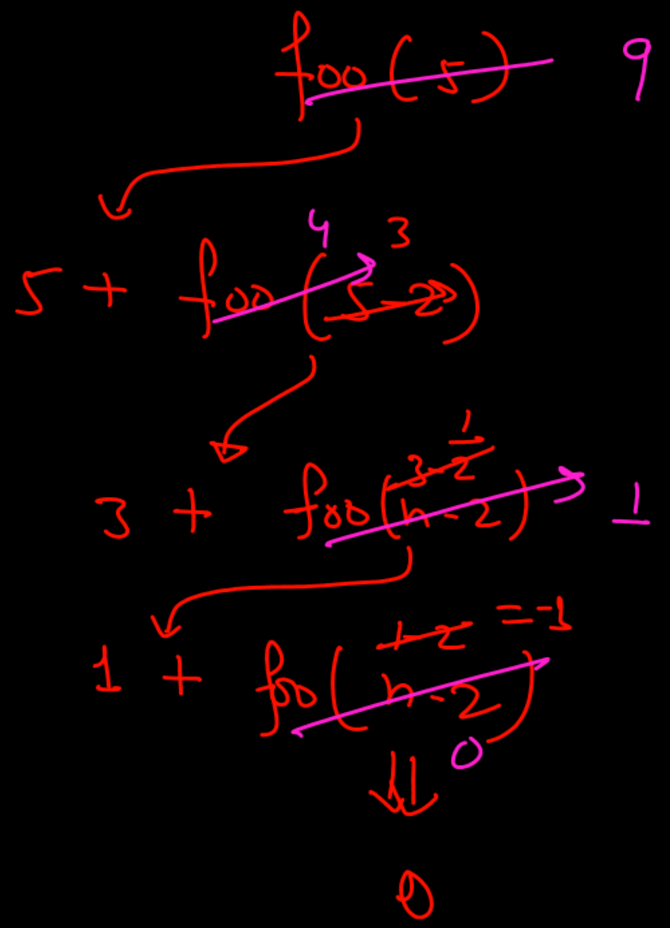
C) 24

D) Undefined behavior



Consider the following recursive function. What does it return when called with foo(5)?

```
int foo(int n) {
    if (n <= 0)
        return 0;
    else
        return n + foo(n - 2);
}
```



A) 5

B) 9

C) 15

D) Undefined

Which of the following statements about the static keyword in functions is true?

A) A static function can be accessed from other files.

B) Static variables in functions are initialized only once.

C) Static functions do not consume memory until they are called.

D) Static variables in a function are reinitialized with each function call.

What will the output of the following code be?

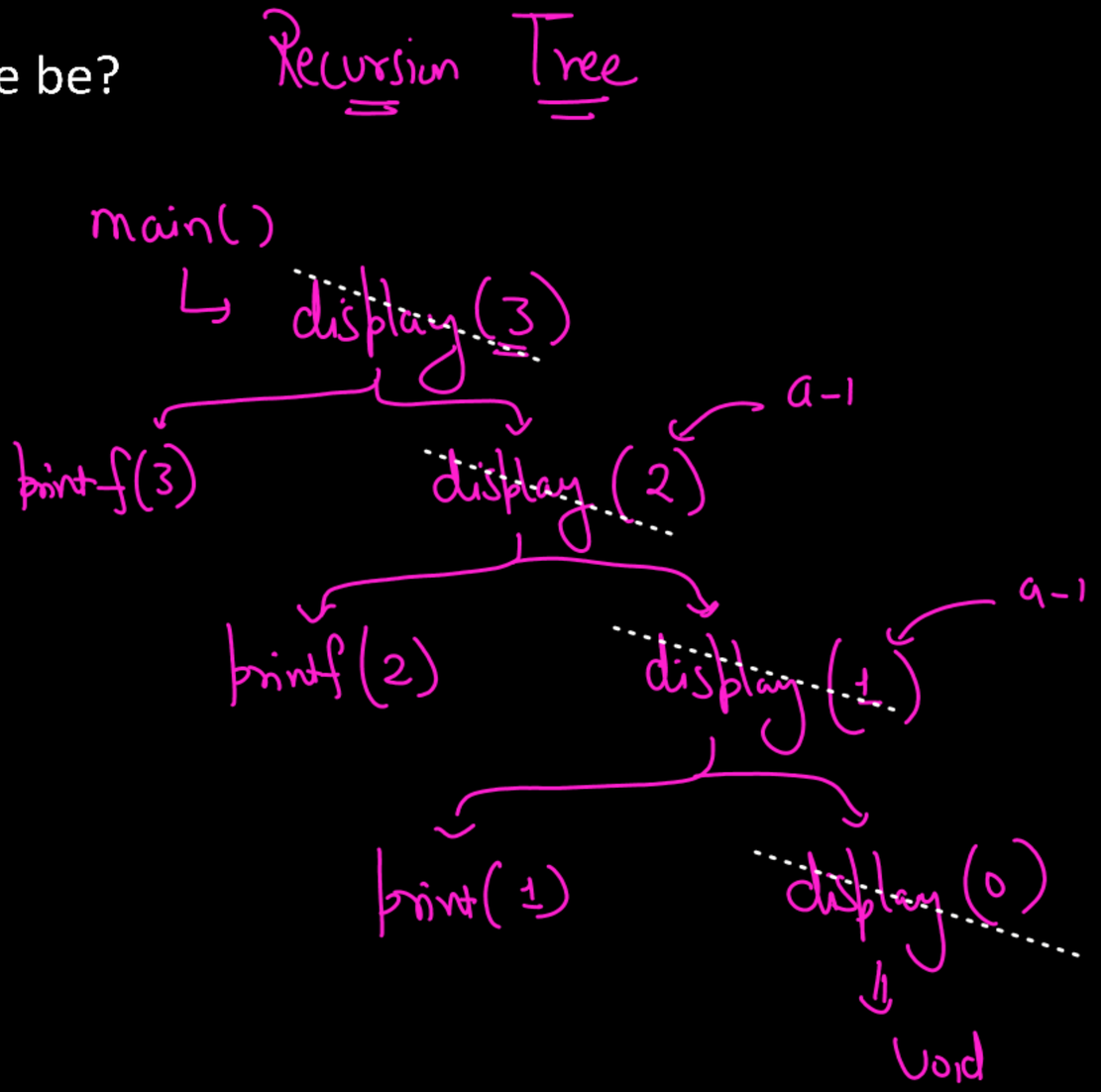
```
#include <stdio.h>
void display(int a) {
    if (a > 0) {
        printf("%d ", a);
        display(a - 1);
    }
}
int main() {
    display(3);
    return 0;
}
```

A) 3 2 1

B) 1 2 3

C) 0 1 2 3

D) Infinite recursion



If a function in C does not have an explicit return type, what is assumed?

A) int ← ANSI

B) void

C) float

D) Compiler error

Which of the following best describes the behavior of inline functions in C?

A) They are expanded at compile time to improve runtime performance.

B) They increase the size of the code by storing multiple copies.

C) They are always stored in a separate memory location.

D) They can only be used once in a program.

What is the output of this code snippet?

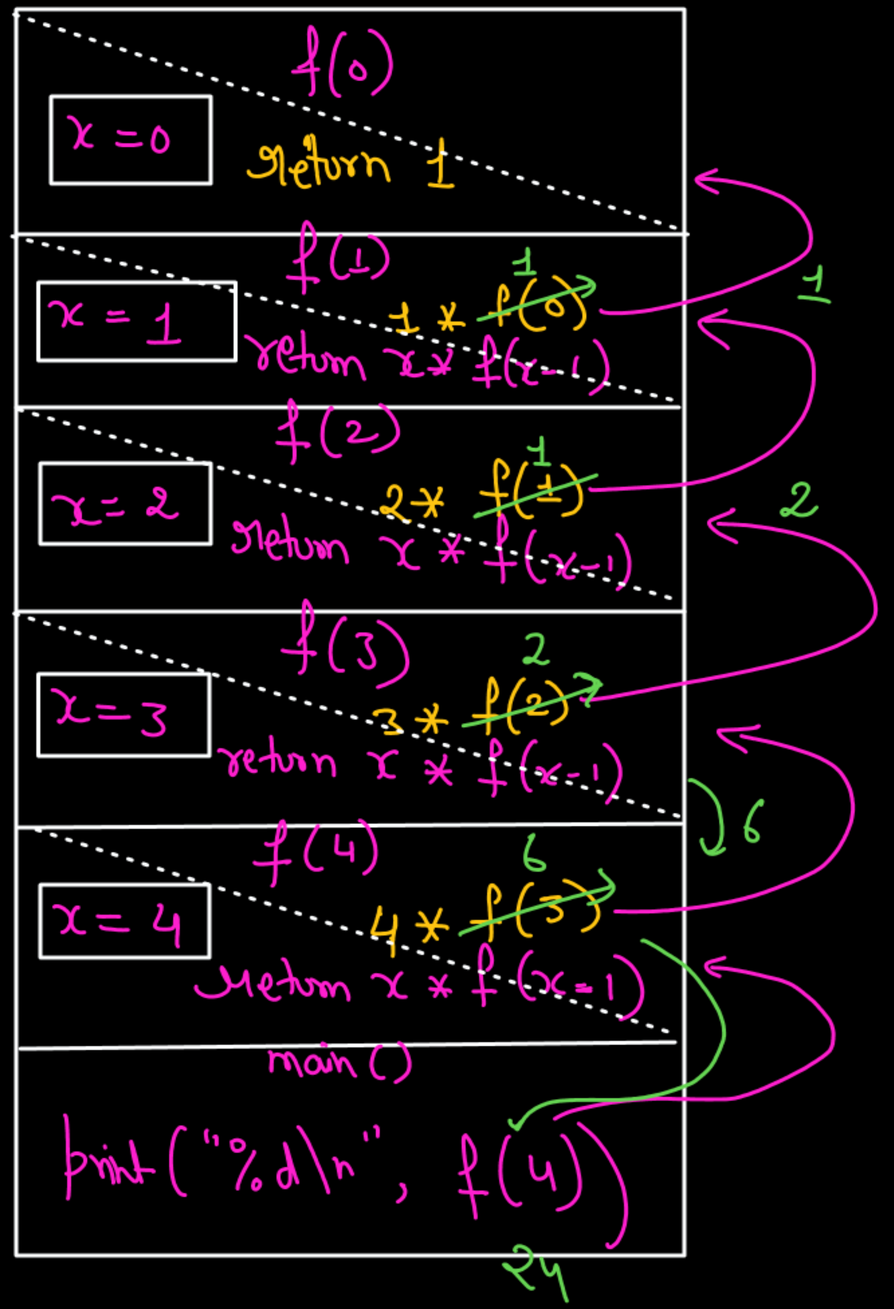
```
#include <stdio.h>
int f(int x) {
    if (x == 0)
        return 1;
    return x * f(x - 1);
}
int main() {
    printf("%d\n", f(4));
    return 0;
}
```

A) 0

B) 4

C) 24

D) 120



What happens when a local static variable in a function is modified?

a) Its new value persists across function calls.

b) Its value resets to default.

c) It causes undefined behavior.

d) It cannot be modified.

When would an inline function benefit performance most?

a) For very large functions.

b) For very small functions called frequently.

c) Inline functions always worsen performance.

d) When used with global variables.

If a global variable is declared static, where can it be accessed?

- a) Within the same function only.
- b) Within the same file.
- c) Anywhere in the program.
- d) It is accessible in all linked files.

Static Keyword used with a variable or a function then its Scope will limit to same file only.

What will be the scope of a variable if it is declared with static inside a function?

- a) The entire program.
- b) The entire file.
- c) Only within the function.
- d) Within the function and its sub-functions.

```
Void fun(){
    static int x=20;
}
```

Scope? ← only within the function

In a nested function call, where is each function's data stored?

a) Stack

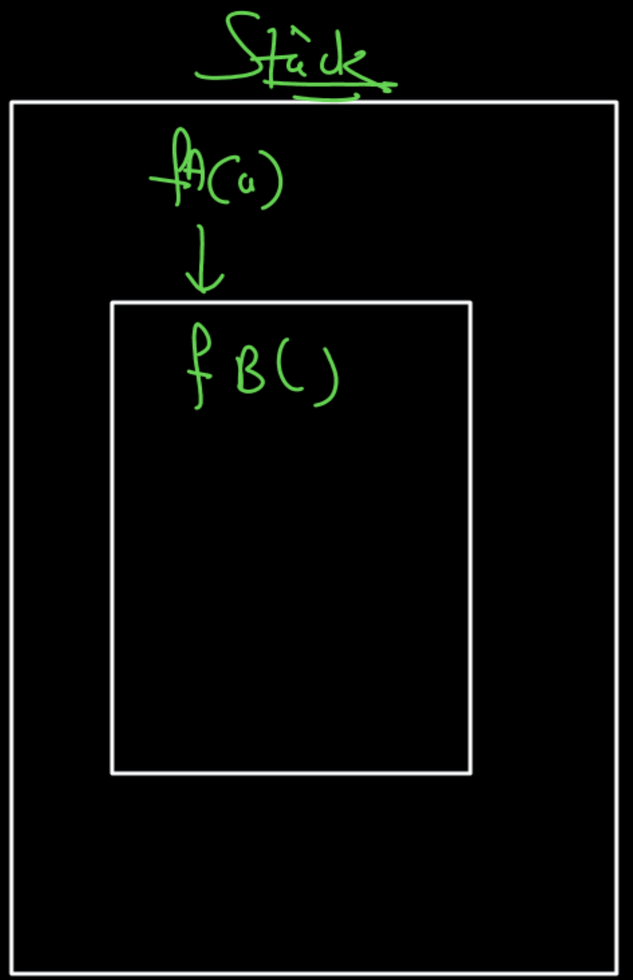
b) Heap

c) Global memory

d) CPU registers

local variables

Request



In which scenario would a static variable inside a recursive function retain different values across recursive calls?

a) If a register variable is also used.

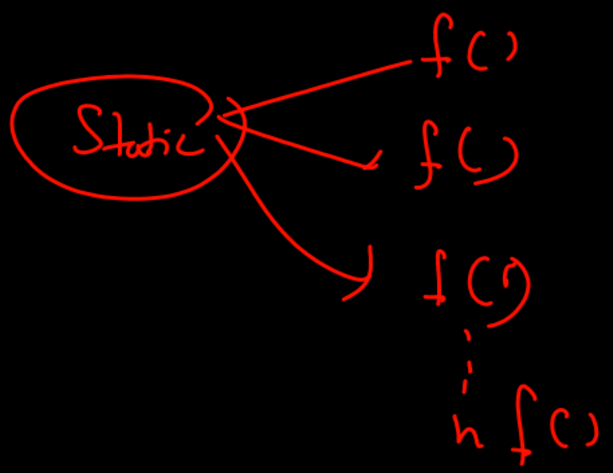
b) If recursion involves different instances of the function.

c) If auto storage class is used.

~~d) This is impossible.~~

f A( ) ⟶ Static int m = 10;
↳ m++
return f(A) ←

Static Variable
⇓
Created only once
↓
Data Segment



In the code below, what will be printed

```c
#include <stdio.h>
int func() {
    static int count = 5;
    return count--;
}
int main() {
    for (int i = 0; i < 5; i++) {
        printf("%d ", func());
    }
    return 0;
}
```

a) 5 5 5 5 5

b) 5 4 3 2 1

c) 5 4 3 2 0

d) Undefined behavior

Static count = 5 4 3 2 1 0

main ( )

↳ for → i = 0, 1, 2, 3, 4

printf ("%d", func ())

i = 0, func ()

i = 1, func ()

i = 2, func ()

i = 3, func ()

i = 4, func ()

5 ← 5 times

output

5 4 3 2 1

Given this code snippet, what is the expected output

```
#include <stdio.h>
int x = 0;
void func() {
    static int x = 5;
    if (x-- > 3) {
        printf("%d ", x);
        func();
    }
}
int main() {
    func();
    return 0;
}
```

Static
int x = 5 4 3 2

a) 4 3 2 1 0
b) 4 3
c) 4 3 2
d) Infinite loop

output:
4 3