DPP ON STRUCTURES AND UNIONS IN ‘C’

LECTURE 42

PROGRAMMING IN ‘C’

Structure

↓

```
struct student {
    int rollNo; ← 4
    float marks; ← 4
     char name[20]; ← 20
};
```

total Size = 28Bytes

Unions ← Same as structure

↓

memory Efficient

```
union student {
    int rollNo; ← 4
    float marks; ← 4
     char name[20]; ← 20
};
```

total Size: 20 Bytes

↳ max (size of each datatype)

```
Struct employee {
  int empid ; ← 4B
  float salary ; ← 4B
  int time ; ← 4B
}
```

→ Size of (Struct employee)

⇒ 12 Bytes

```
union employee {
  int empid ; ← 4
  float salary ; ← 4
  int time ; ← 4
}
```

→ Sizeof (union employee)

⇒ 4 Byte

What is the correct syntax for declaring a structure in C?

A) struct { int ✓a; float ✓b; }; error

B) ✗structure { int a; float b; };

[✓ C)] struct { int a; float b; } myStruct;✓

D) ✗structure { int a, float b } myStruct;

Which of the following is a valid way to define a union in C?

A) `union { int a; float b; };` ✗ error

[✓ B)] `union myUnion { int a, float b; };`

C) `structure myUnion { int a; float b; };` ✗

D) `struct union myUnion { int a, float b; };` ✗

What is the size of a structure that contains an int, float, and a char?

↓ ↓ ↓

4 4 1 = 9 Bytes

ANSI 2 4 1 = 7

A) 4 bytes

B) 8 bytes

C) 12 bytes

[✓ D)] It depends on the compiler

What is the primary difference between a structure and a union in C?

A) A structure stores multiple variables of different data types, while a union stores multiple variables of the same data type. ✗

[✓ B) A structure allocates memory for all its members, while a union allocates memory for the largest member only.]

C) A structure can only store integers, while a union can store multiple data types. ✗

D) There is no difference; they are the same. ✗

Which of the following is the correct way to access the float member b of a structure s?

s.b

← member access operator

[✓ A)] `s.b`

B) `s->b`

C) `b.s`

D) `->s.b`

What is the output of the following code?

```
union myUnion {
    int x;
    float y;
};
int main() {
    union myUnion u;
    u.x = 10;
    u.y = 20.5;
    printf("%d %f", u.x, u.y);
    return 0;
}
```

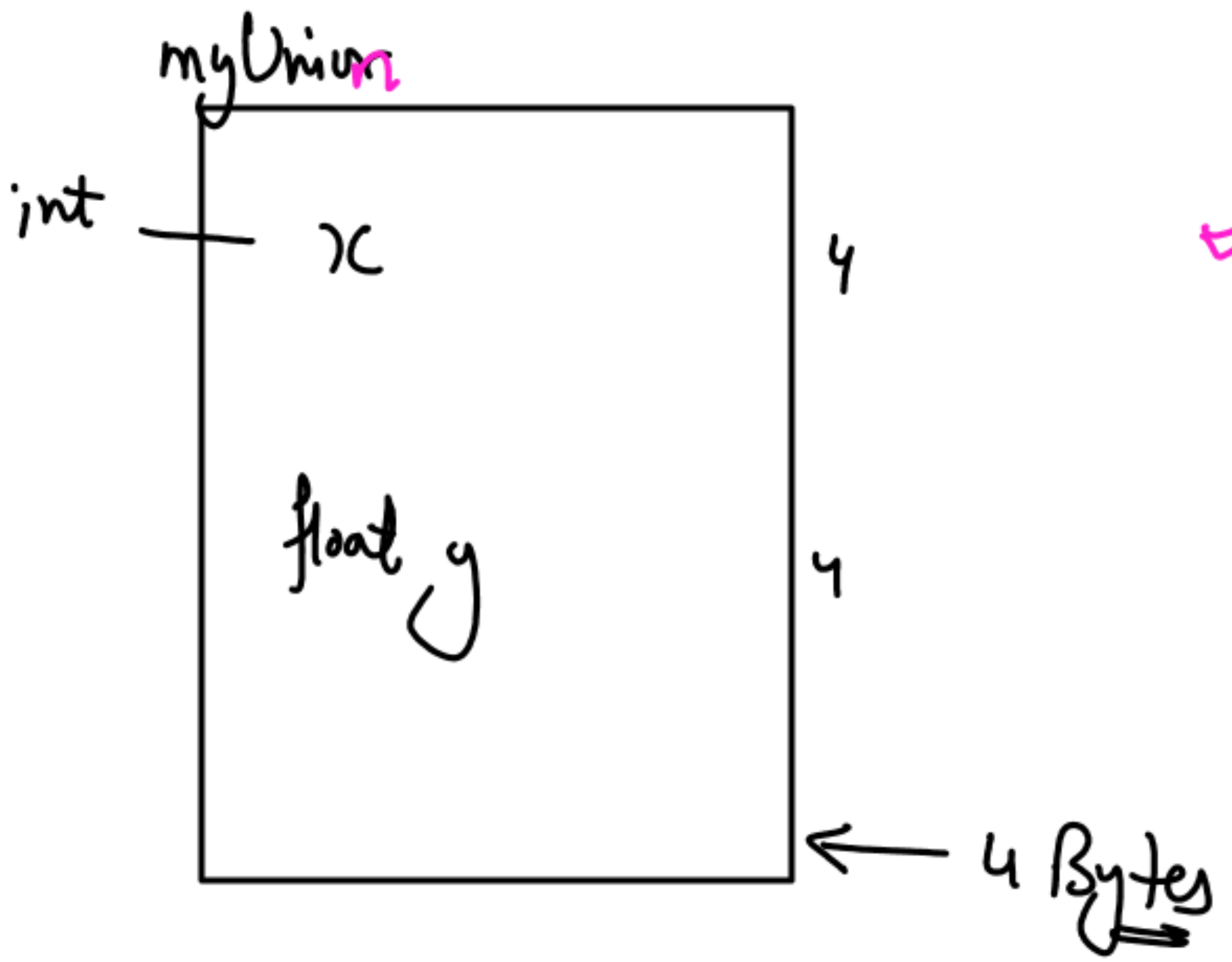
↓ 0 ↓ 20.5

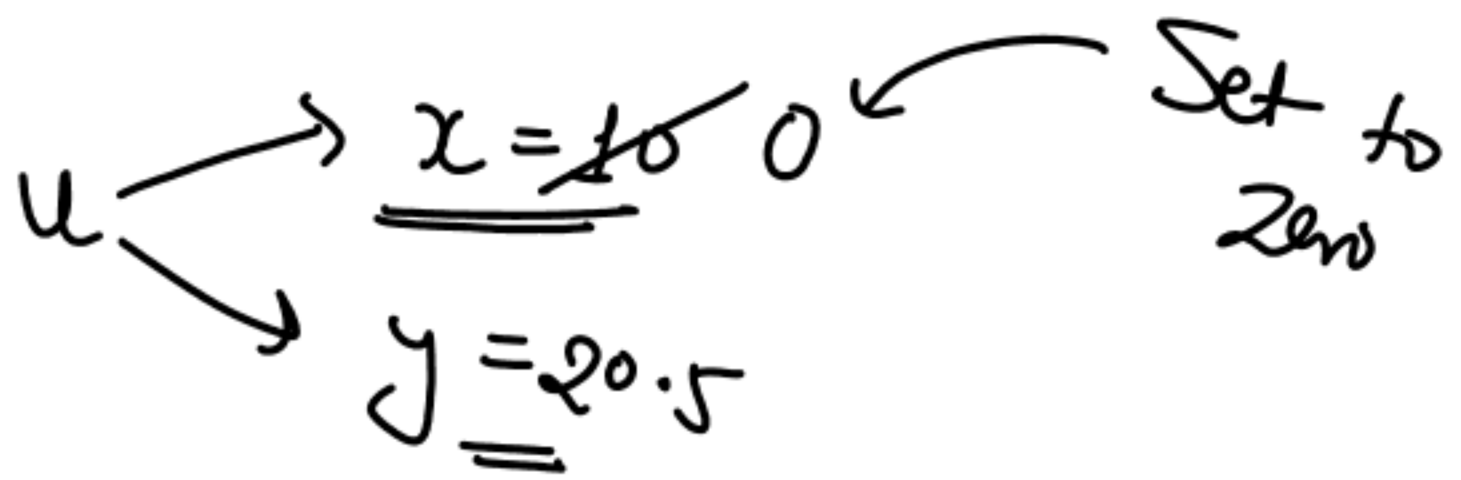
A) 10 20.5

B) 10 0.0

[✓ C) 0 20.5]

D) Undefined output





What is the correct syntax to declare a pointer to a structure?

[✓ A)] `struct myStruct *ptr;`

B) `struct *ptr;`

C) `pointer struct myStruct;`

D) `struct myStruct ptr;`

int * ptr

Struct Structure * ptr

How does memory allocation work in a union?

A) Memory is allocated for each member separately.

[✓ B) Memory is allocated for the largest member only.]

C) Memory is allocated for all members equally.

D) No memory is allocated for a union.

What is the size of the following union?

```
union myUnion {

 char c; ←  1 Byte

 int i; ←  4 Bytes

double d; ←  8byte

};
```

A) 1 byte

B) 4 bytes

[✓ C) 8 bytes]

D) 16 bytes

Consider the following code:

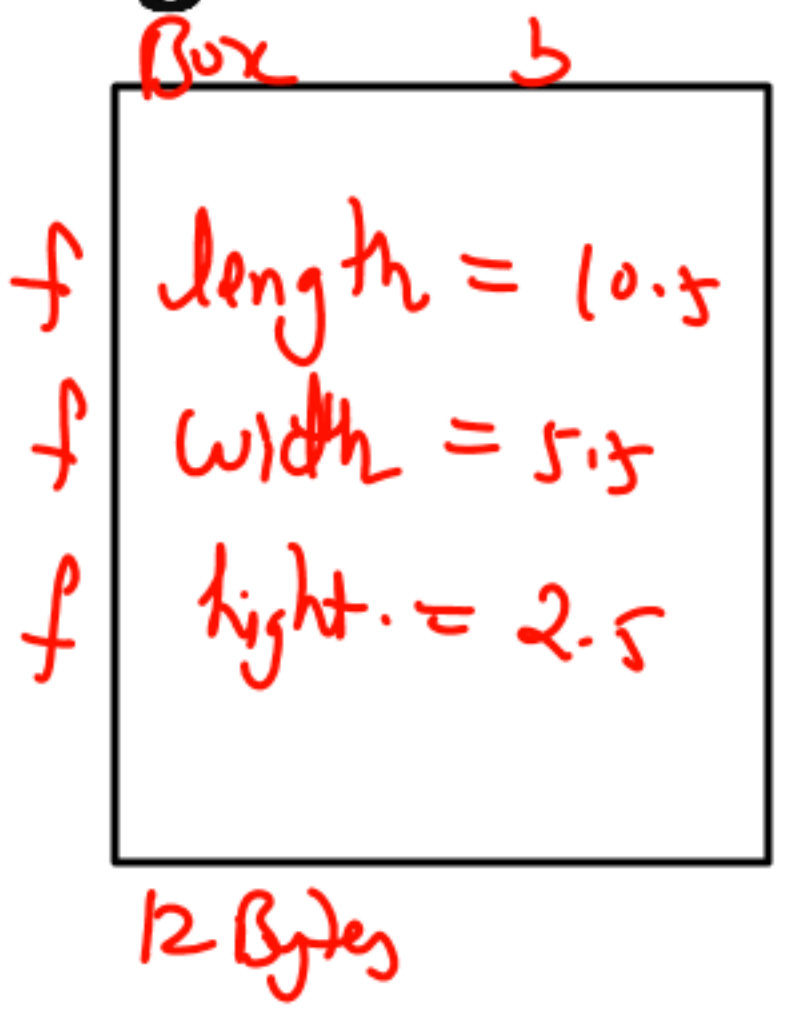
```
#include <stdio.h>
struct Box {
    float length;
    float width;
    float height;
};
union Value {
    int integer;
    float decimal;
};
int main() {
    struct Box b = {10.5, 5.5, 2.5};
    union Value v;
    v.integer = 100;
    printf("Box dimensions: %.2f %.2f %.2f\n", b.length, b.width, b.height);
    printf("Union integer: %d\n", v.integer);
    v.decimal = 3.14;
    printf("Union decimal: %.2f\n", v.decimal);
    printf("Union integer after decimal assignment: %d\n", v.integer);
    return 0;
}
```

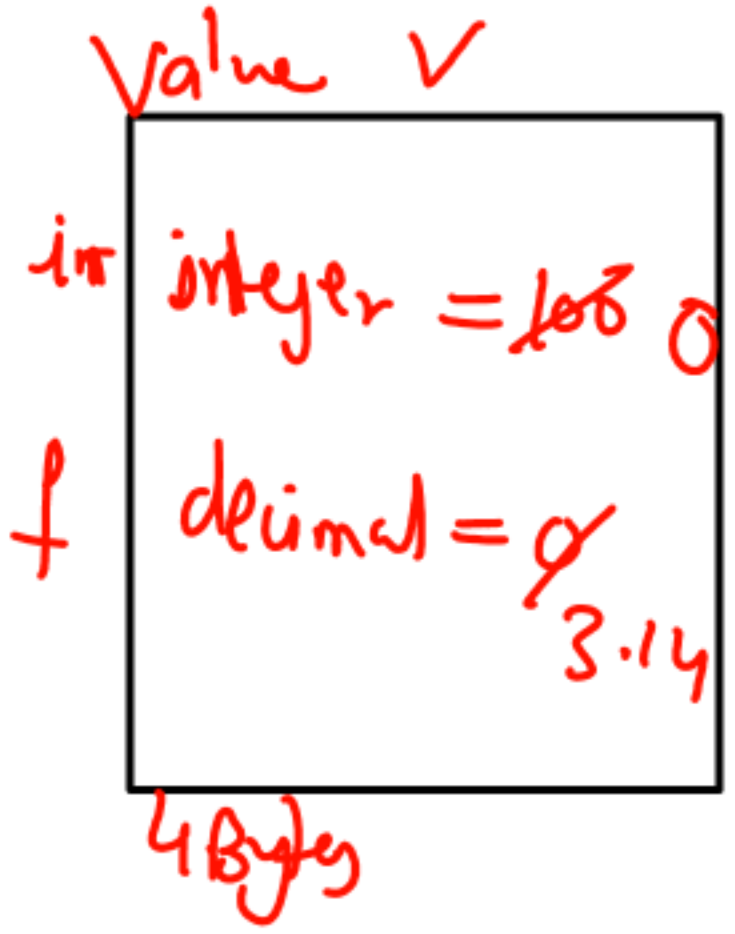
10.50 5.50 2.50

100

3.14

0





A) ✓
Box dimensions: 10.50 5.50 2.50 ✓
Union integer: 100 ✓
Union decimal: 3.14 ✓
Union integer after decimal assignment: 0 ✓

B)
Box dimensions: 10.50 5.50 2.50
Union integer: 100
Union decimal: 3.14
Union integer after decimal assignment: 3

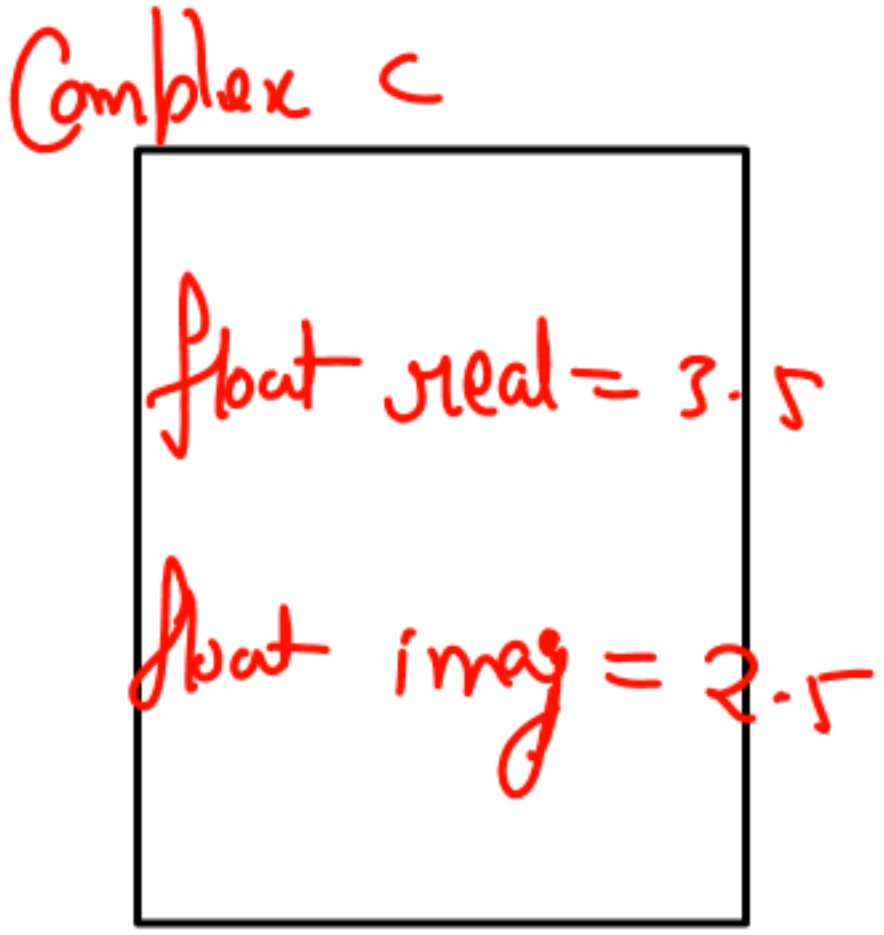
C)
Box dimensions: 10.50 5.50 2.50
Union integer: 100
Union decimal: 3.14
Union integer after decimal assignment: 100

D)
Box dimensions: 10.50 5.50 2.50
Union integer: 0
Union decimal: 3.14
Union integer after decimal assignment: 0

What will be the output of the following code?

```
#include <stdio.h>

struct Complex {

    float real;

    float imag;

};

int main() {

    struct Complex c = {3.5, 2.5};

    printf("%.2f + %.2fi", c.real, c.imag);

    return 0;

}
```

3.50 + 2.50;


Complex c

float real = 3.5

float imag = 2.5

✓ A) 3.50 + 2.50i

B) 3.5 + 2i

C) 0.0 + 0.0i

D) Undefined

What will be the output of the following code?

```
#include <stdio.h>
struct A {
    int x;
    char y;
};
struct B {
    struct A a;
    float z;
};
int main() {
    struct B b = {{1, 'A'}, 3.14};
    printf("%d %c %.2f", b.a.x, b.a.y, b.z);
    return 0;
}
```

Struct A a
↳ x = 1
y = 'A'

Struct B b
↳ Struct A a
float z;
↳ 3.14

Struct A a = {1, 'A'}

1 'A' 3.14

[✓ A) 1 A 3.14]

B) 1 A 0.00

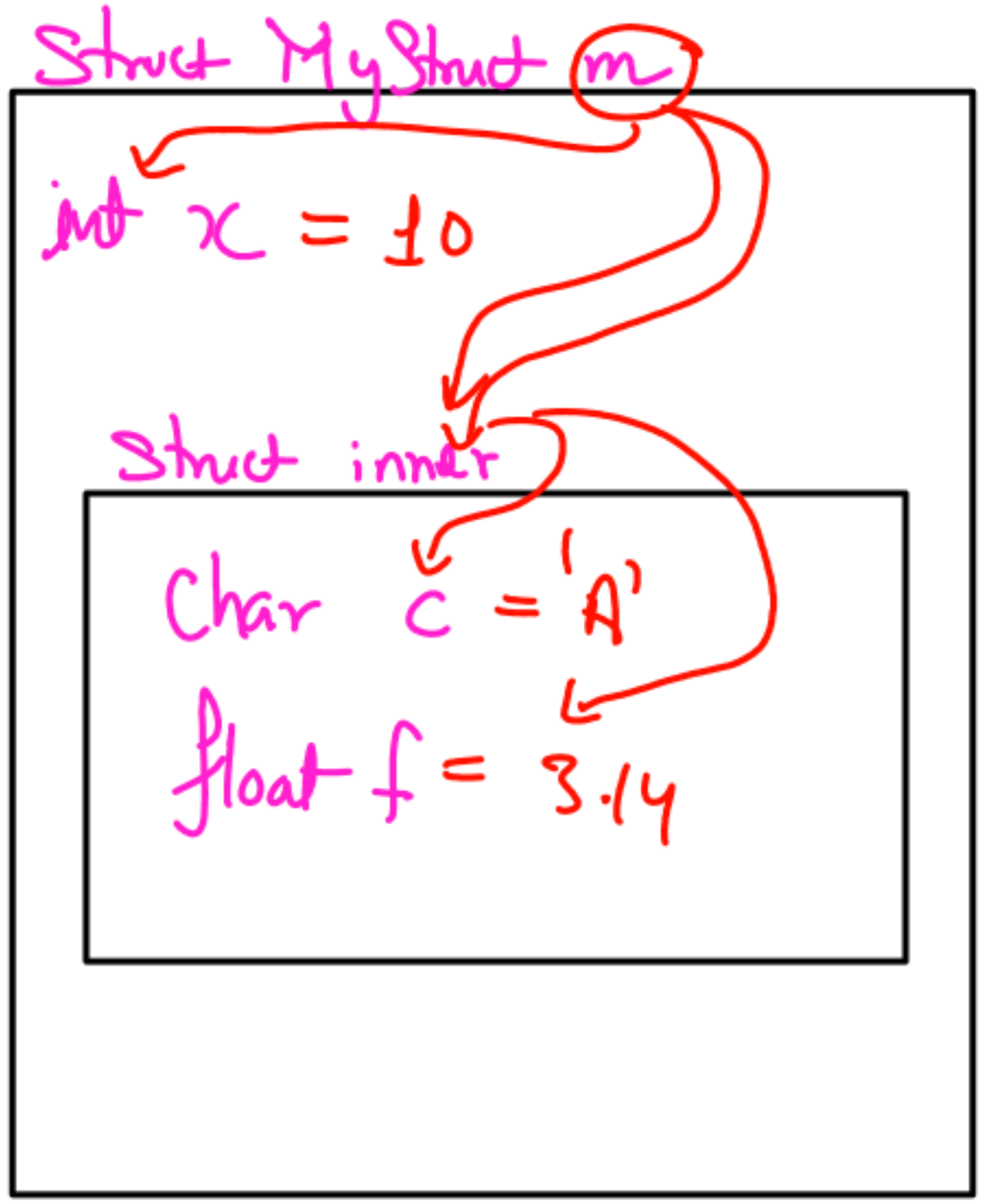
C) 0 0 3.14

D) 1 0 3.14

What will be the output of the following code?

```
#include <stdio.h>
struct MyStruct {
    int x;
    struct {
        char c;
        float f;
    } inner;
};
int main() {
    struct MyStruct m = {10, {'A', 3.14}};
    printf("%d %c %.2f", m.x, m.inner.c, m.inner.f);
    return 0;
}
```

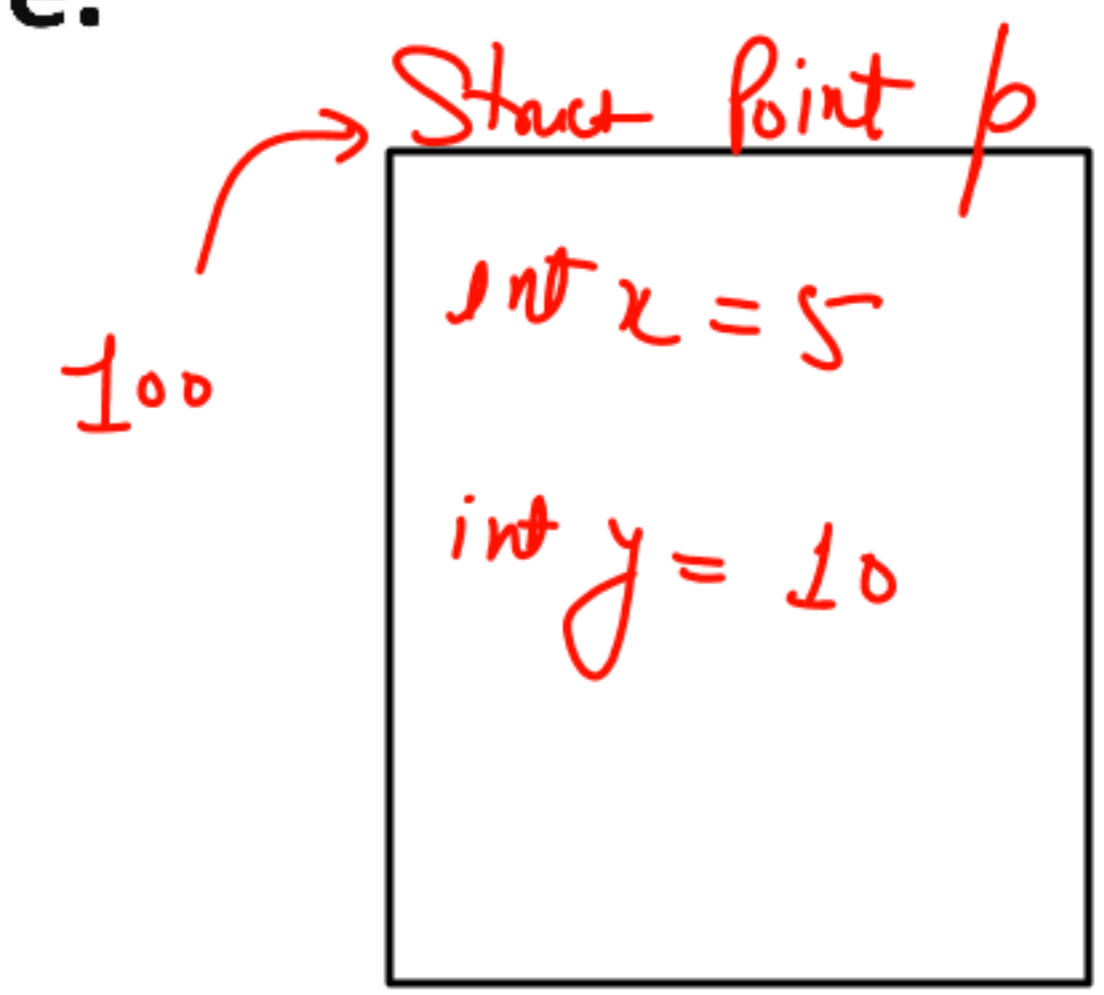
10 'A' 3.14



Consider the following code.

```
#include <stdio.h>

struct Point {

    int x, y;

};

int main() {

    struct Point p = {5, 10};

    struct Point *ptr = &p;

    printf("%d %d\n", ptr->x, ptr->y);

    return 0;

}
```

5 10



What will be the output?

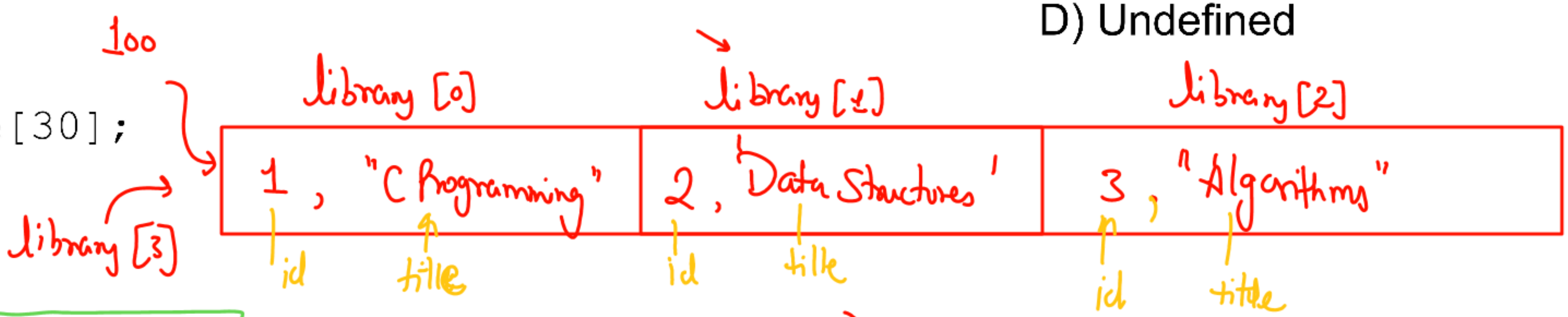
[✓ A) 5 10]
B) 10 5
C) x y
D) Undefined

pointer ptr = &p

What will be the output of the following code

```
#include <stdio.h>
struct Book {
    int id;
    char title[30];
};
int main() {
 struct Book library[3] = {{1, "C Programming"}, {2,
 "Data Structures"}, {3, "Algorithms"}};
 struct Book (*ptr)[3] = &library;
 printf("%d %s\n", (*ptr)[1].id, (*ptr)[1].title);
 return 0;
}
```

[✓ A) 2 Data Structures]
B) 1 C Programming
C) 3 Algorithms
D) Undefined



ptr = & library

library [1]

"2 DataStructure"

Pointing the array of 3 elements

Consider the following C declaration

```
struct node {
    int i; ✓
    float j; ✓
};
struct node *s[10];
```

← Pointer (Struct Node) → Pointing the array of Structure holding 10 elements

This defines:

(a) An array, each element of which is a pointer to a structure of type node

✓ (b) A structure of 2 fields, each field being a pointer to an array of 10 elements .

(c) A structure of 3 fields: an integer, a float, and an array of 10 elements

(d) An array, each element of which is a structure of type node

Assume that objects of type short, float, and long occupy 2 bytes, 4 bytes, and 8 bytes respectively. Consider the following declaration:

```
struct {
    short s[5]; ← 5X2 = 10 Bytes
    union {
        float y; ← 4 Bytes
        long z; ← 8 Bytes
    } u; ← 8 Bytes
} t; ←
```

The memory requirement for variable t, ignoring alignment considerations, is:

(a) 22 bytes

[✓ (b) 18 bytes]

(c) 14 bytes

(d) 10 bytes

Consider the following C program

```
void f(int, short);
void main() {
    int i = 100;
    short s = 12;
    short *p = &s;
    __________ ; // call to f()
}
```

Which one of the following expressions, when placed in the blank above, will NOT result in a type-checking error?

(a) f(s, *s); ✗

(b) i = f(i, s); error

(c) f(i, *s); ← error

[✓ (d)] f(i, *p);

Given the code snippet:

```
struct Test {

    int x;              x=5

    char y;             y='A'

    double z;→'0'

};

struct Test t = {5, 'A'};
```

What will happen during compilation?

[✓ a)] Compiles successfully with z initialized to 0 ✓

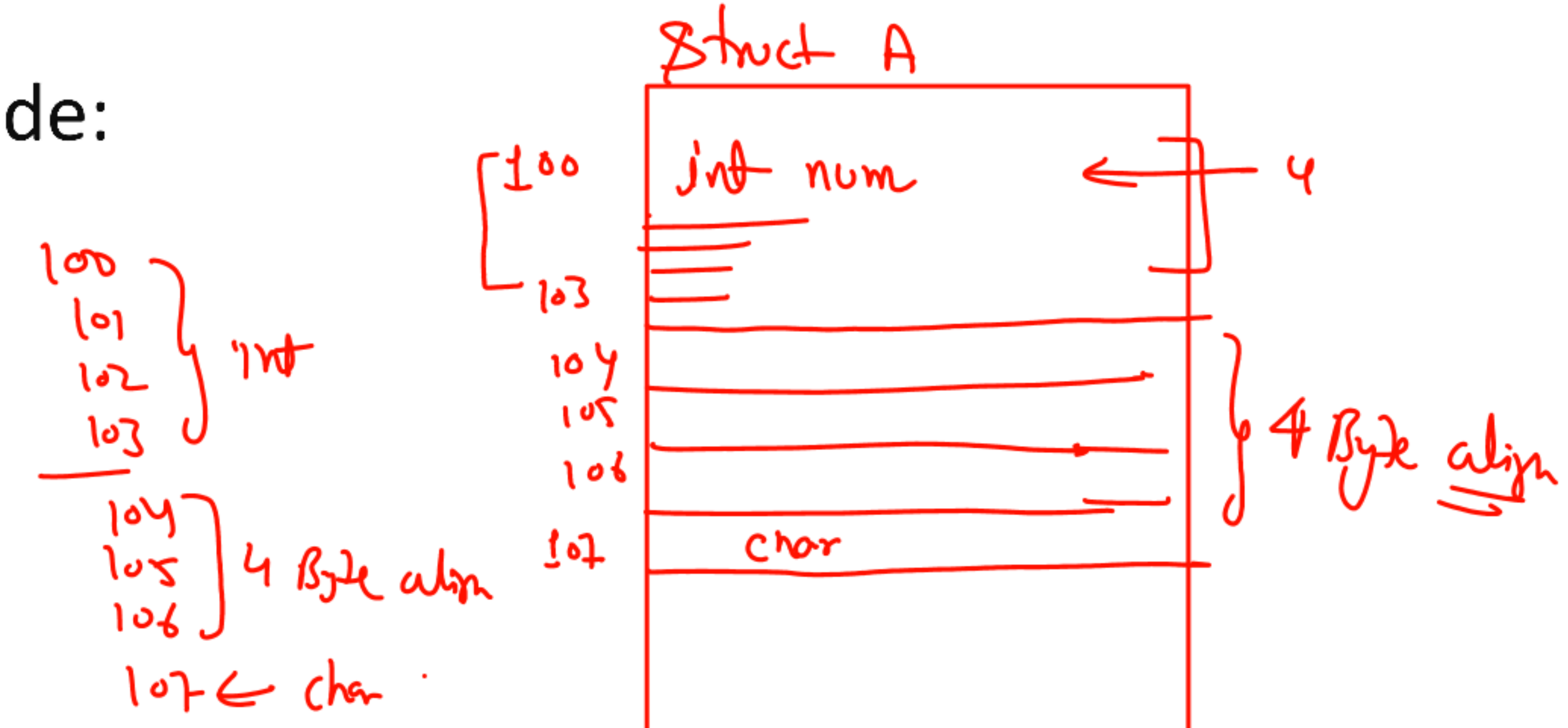
b) Syntax error: Missing initializer for z

c) Undefined behavior at runtime

d) Syntax error: Structure requires all members to be initialized

Consider the following code:

```
struct A {
    char ch;
    int num;
};
printf("%zu", sizeof(struct A));
```



If sizeof(int) = 4 and sizeof(char) = 1, what is the output on a system with 4-byte alignment?

a) 5

[✓ b) 8]

c) 9

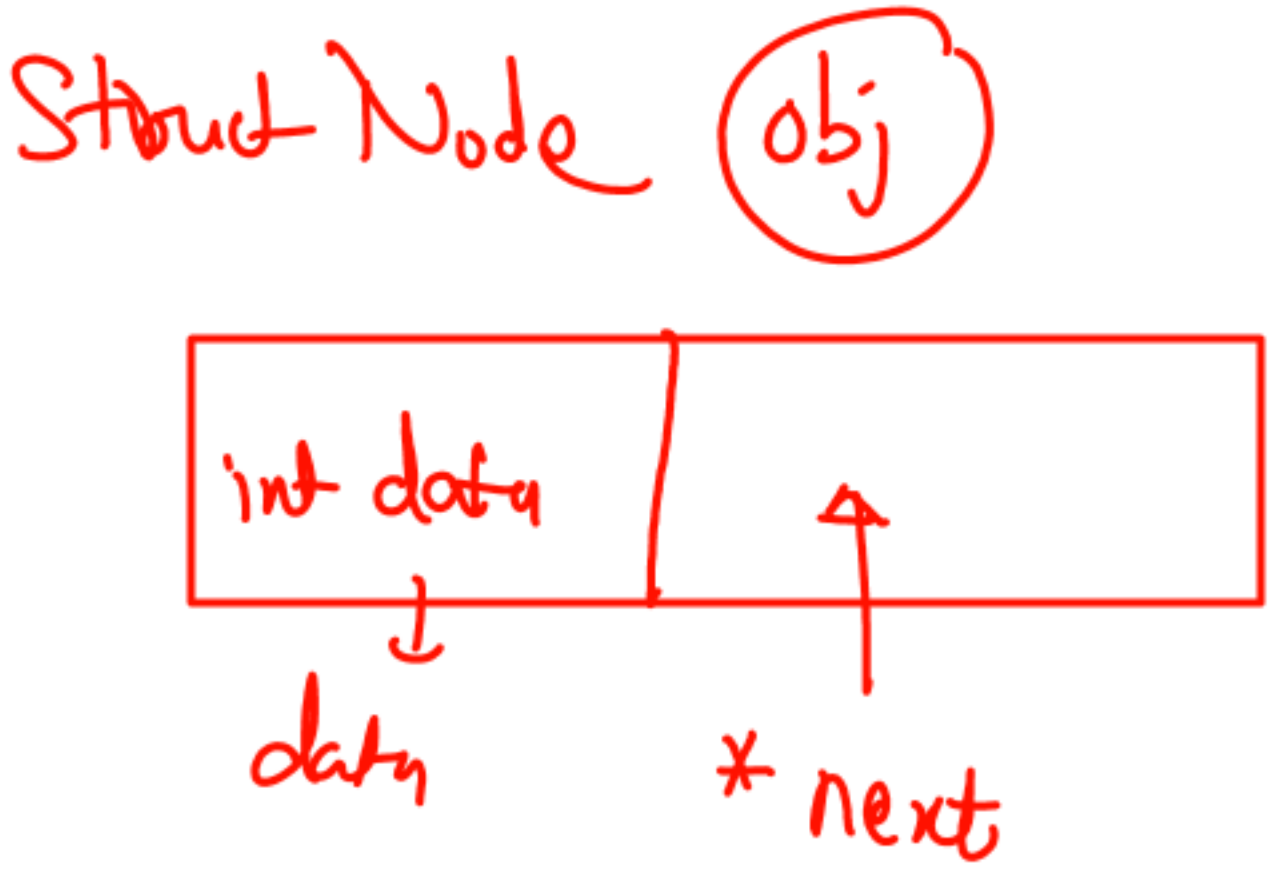
d) 12

int → 4 B

Char → 1

Given:

```
struct Node {
    int data;
    struct Node *next;
};
struct Node obj;
printf("%p", &(obj.next));
```



What does &(obj.next) point to?

a) The address of obj

[✓ b) Address immediately after obj.data]

c) Address of the next Node

d) Undefined behavior

For the following union:

```
union Example {

    int a; ← 4

    double b; ← 8

    char c; ← 1

};

printf("%zu", sizeof(union Example));
```

Assume sizeof(int) = 4, sizeof(double) = 8, and sizeof(char) = 1. What is the output?

[✓ a) 8]

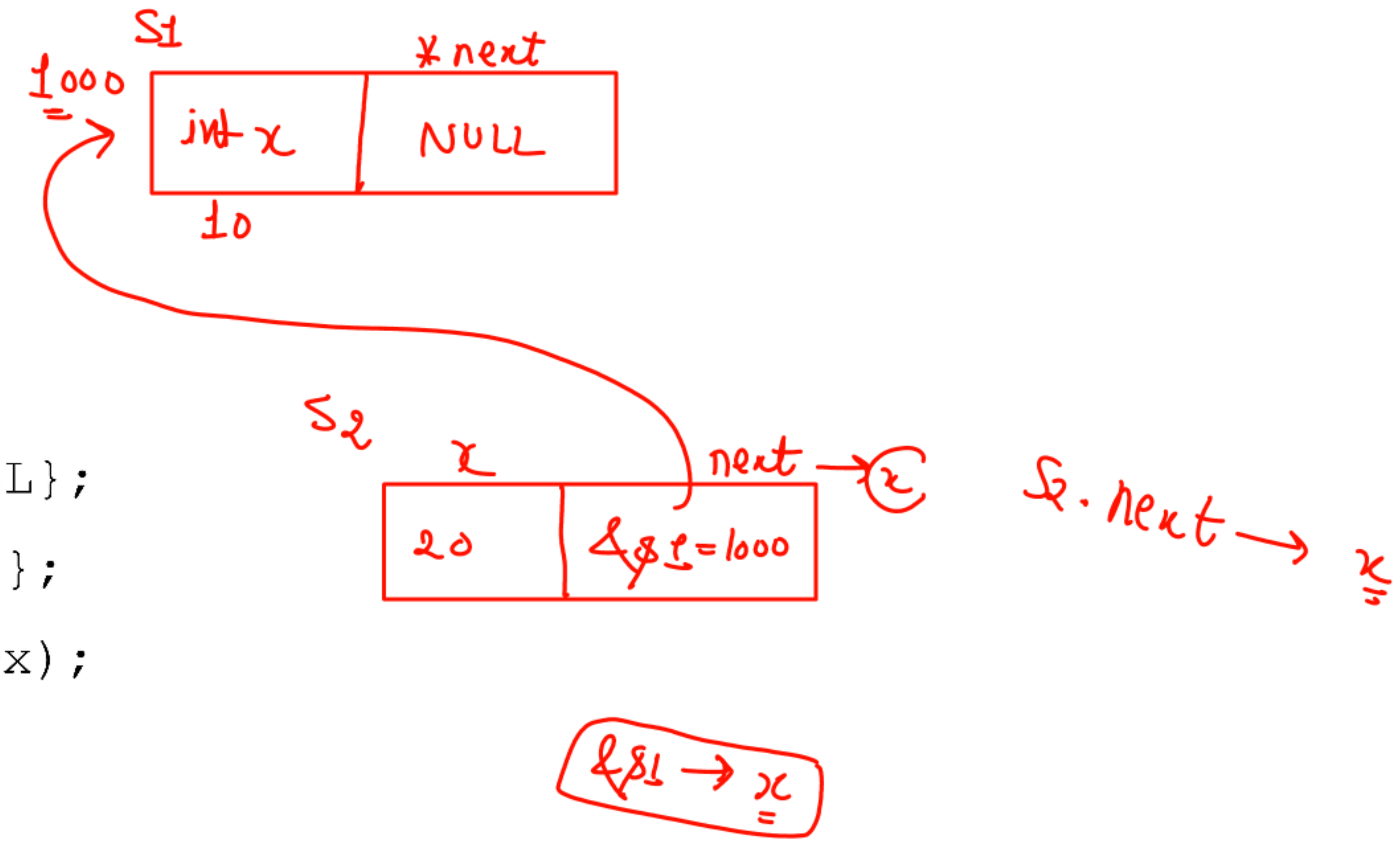
b) 9

c) 16

d) 12

Consider:

```
struct S {
    int x;
    struct S *next;
};
struct S s1 = {10, NULL};
struct S s2 = {20, &s1};
printf("%d", s2.next->x);
```



What is the output?

[✓ a) 10]

b) 20

c) NULL

d) Compilation error

Which operation is invalid for structures?

a) Copying using = operator

[✓ b) Comparing using == operator] ← Illegal

c) Accessing members using . operator

d) Passing as function arguments

Given the code:

```
struct Point {
    int x, y;
};
struct Point p1 = {10, 20};
struct Point *ptr = &p1;
printf("%d %d", ptr->x, (*ptr).y);
```

P1

x=10
y=20

ptr = &P1

10 20

Value at (ptr). y

P1 . y

What is the output?

a) 10 20

b) 20 10

c) Syntax error: -> operator not valid for pointers

d) Undefined behavior

Consider:

```
struct Test {
    int id;
    char name[20];
};
struct Test *ptr = NULL;
printf("%d", ptr->id);
```

Test

ptr = NULL

What happens?

a) Prints 0

[✓ b) Segmentation fault] ← When a pointer wants to access the address which is not allowed to access.

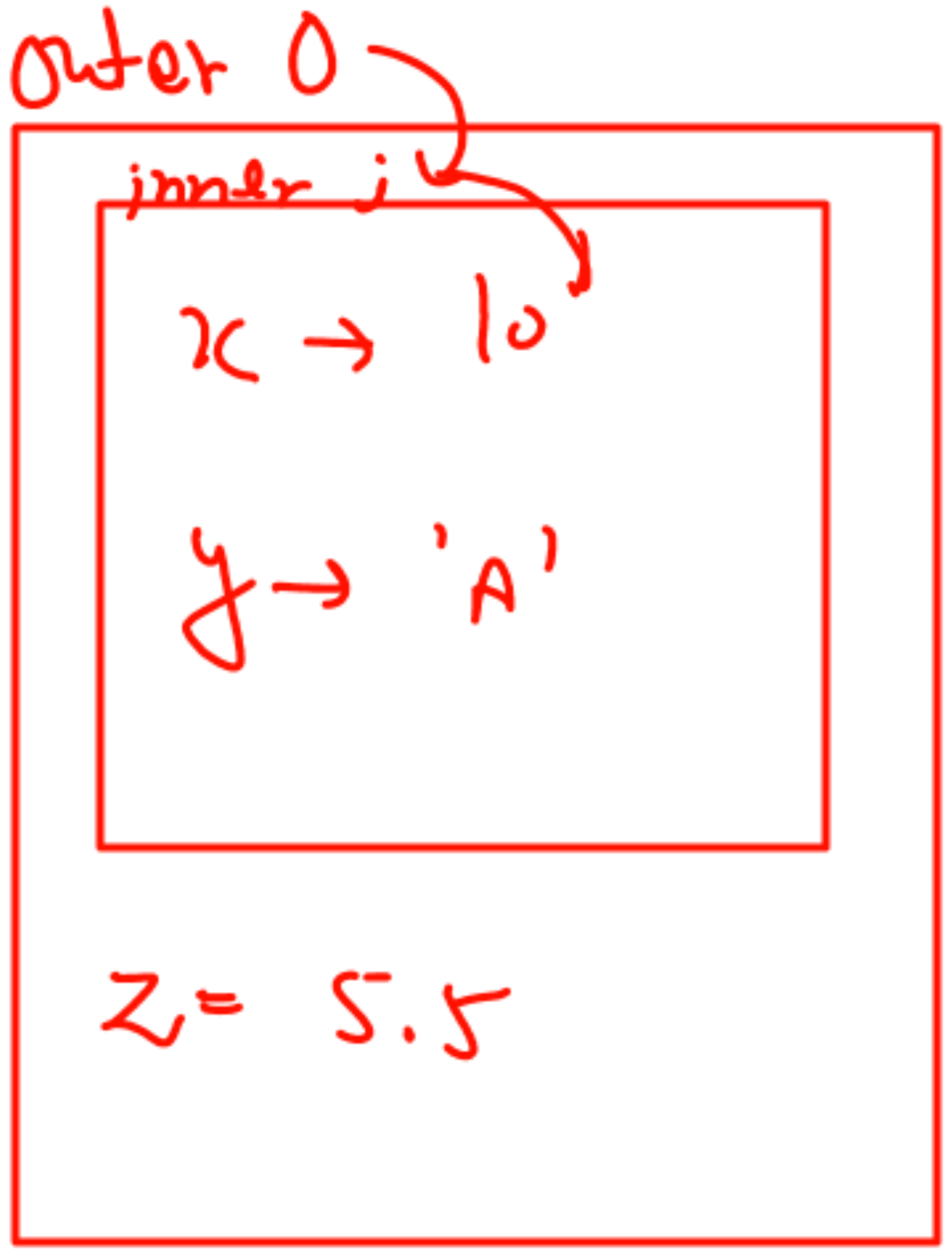
c) Undefined behavior

d) Compilation error

Predict the output:

```
struct Inner {
    int x;
    char y;
};
struct Outer {
    struct Inner i;
    double z;
};
struct Outer o = {{10, 'A'}, 5.5};
printf("%d %c %.1f", o.i.x, o.i.y, o.z);
```

10 'A' 5.5



[✓ a) 10 A 5.5]
b) Garbage A 5.5
c) 0 A 5.5
d) Compilation error

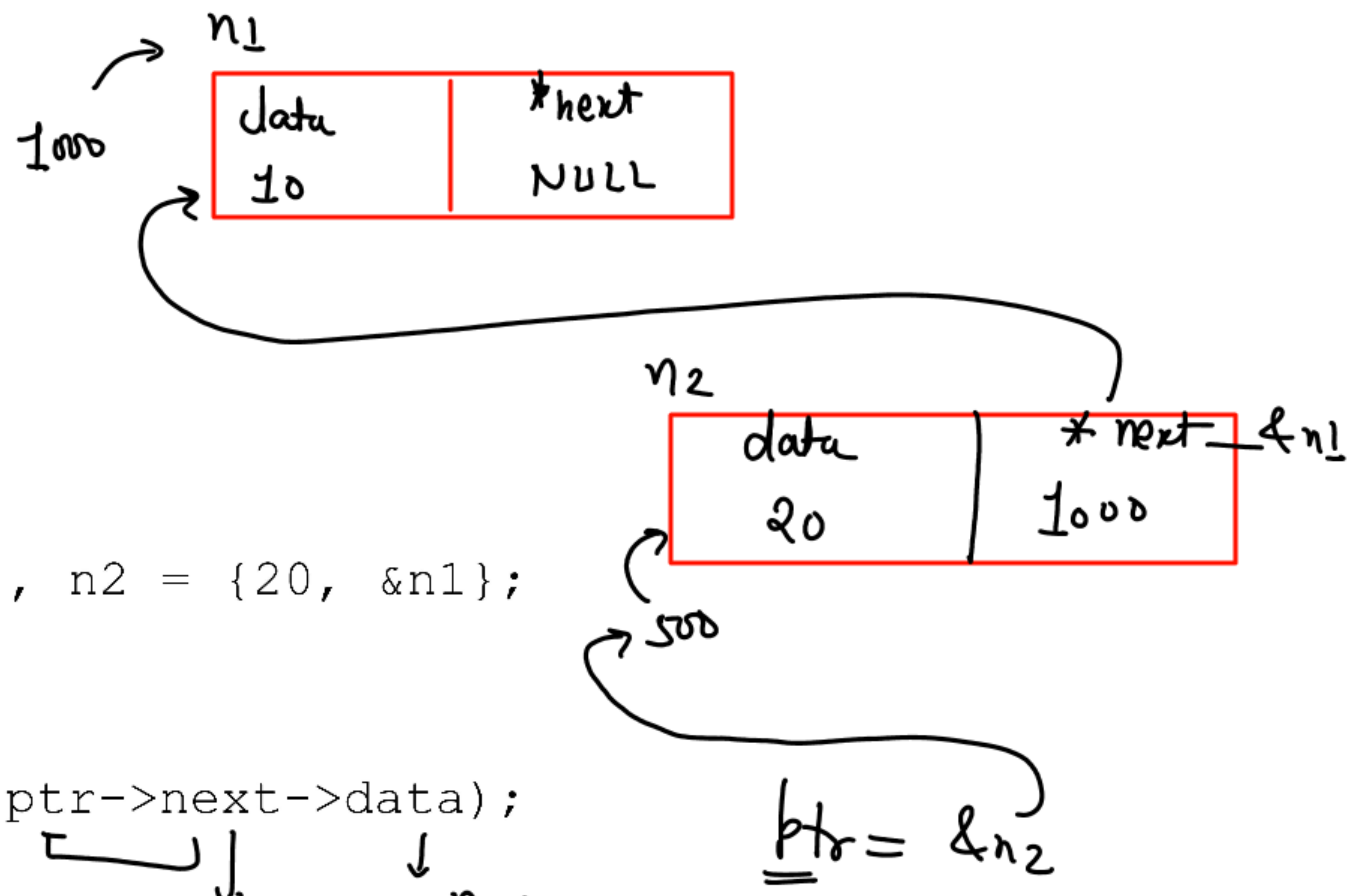
Predict the output:

```
struct Node {
    int data;
    struct Node *next;
};
struct Node n1 = {10, NULL}, n2 = {20, &n1};
struct Node *ptr = &n2;
printf("%d %d", ptr->data, ptr->next->data);
```

20 10

Pointer to the Structure

n1 (data)



[✓ a) 20 10]

b) 10 20

c) 10 NULL

d) Undefined behavior

For a structure pointer:

```
struct A {
    int a;
    float b;
};

struct A *ptr;
```

What does (ptr + 1) point to?

a) Next member of the structure

b) Address after the structure

[✓ c) Address of the next structure in an array]

d) Undefined

int x = 20

int *ptr = &x

ptr + 1

↳ ptr + i x sizeof(int)

DPP

Proj- in C

⇒ DMA ←

2Lect

4 functions

↳ malloc
calloc
realloc
free

आसान