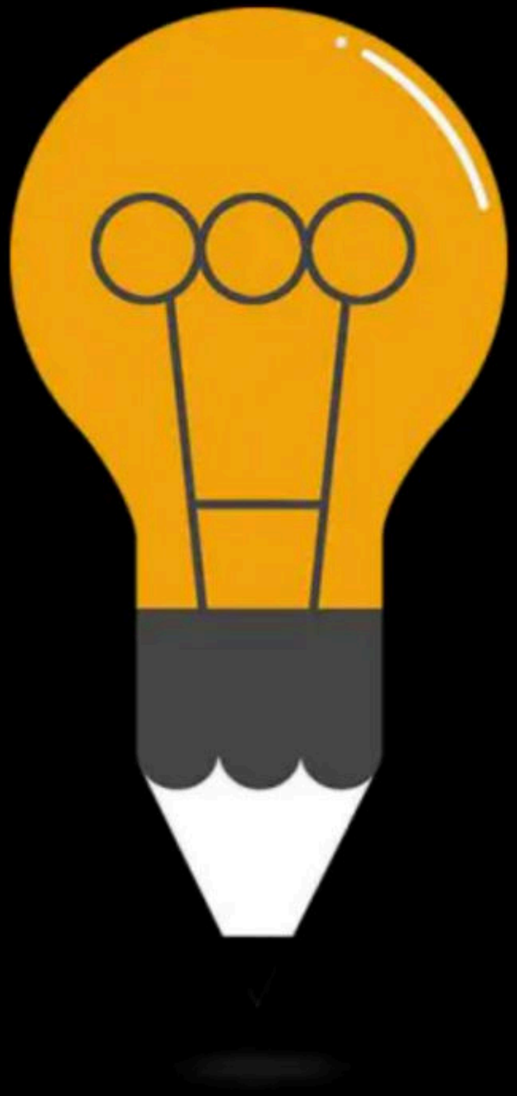


Arrays, Pointers & Structure-Union

By: Vishvadeep Gothi

Question

Ans : 9

```c
#include <stdio.h>

int main() {
    int A[20];
    printf("%d", *A+4-2 -*A+7);
    return 0;
}
```

A ⇒ base add. of array

*A ⇒ element at base address

↓

A[0]

*A+4-2 -*A+7

A[0] + 4 - 2 - A[0] + 7

9

Pointer to array

```
int (*p)[5];
```

p points to an array

(*p)[0]

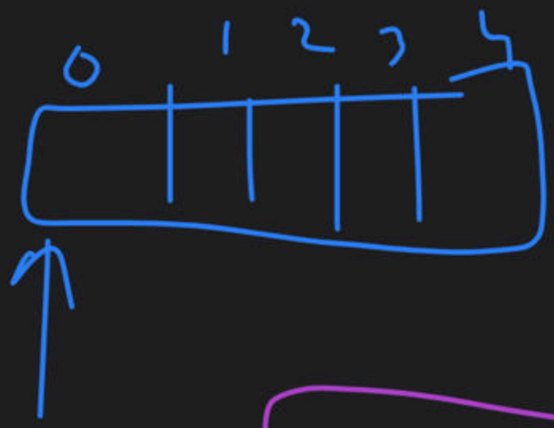
(*p)[1]

(*p)[2]

p = p + 1

↓

inc. done by size of array




```
int *q
```

q points to int

q = q + 1

↓

incremented by size of int

Question

```
int main() {
    int A[7]={7,6,5,4,3,2,1};
    int (*p)[3]=A;
    printf("%d",(*p)[0]);
    p=p+1;
    printf("%d",(*p)[0]);
}
```

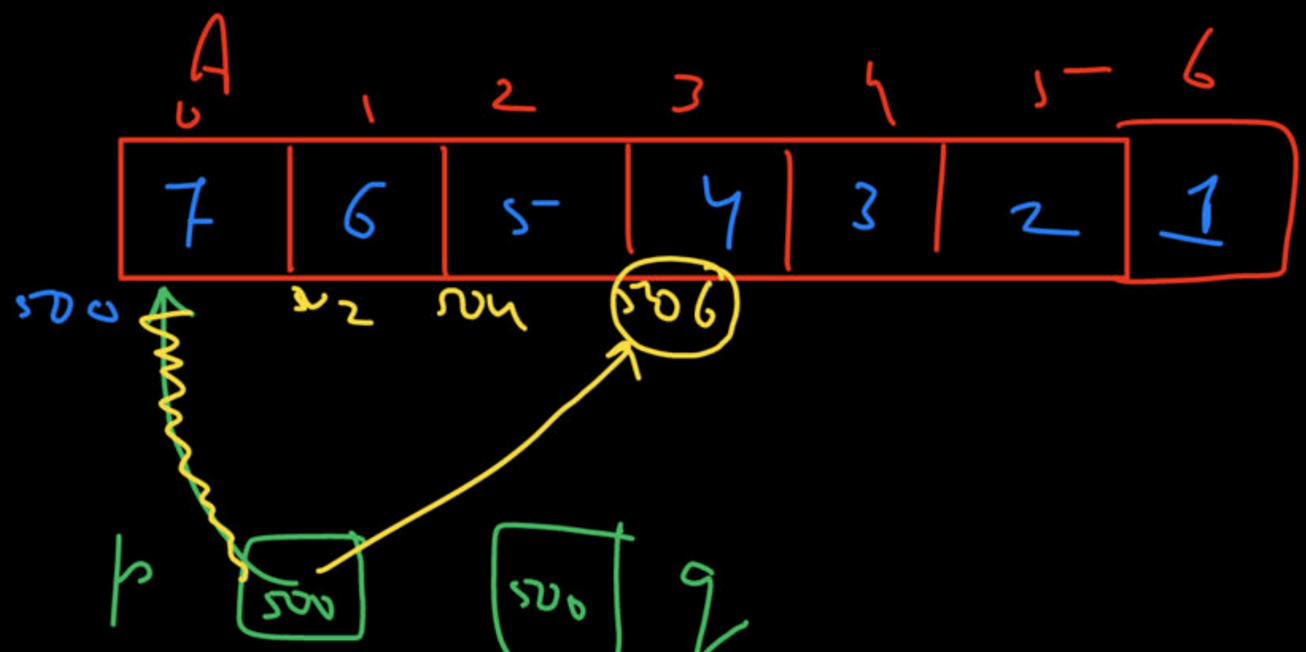
int *q = A;

7

4

q = q + 1;

printf("%d", q[0]); 6



(size 3 element)

p = p + 1

p = 506

q = q + 1

q = 502

Initialization of 2-D Array

1-d array :-

```
int A[5] = { 5, 8, 2, 6, 9 };
```

2-d array :-

```
int A[3][4] = {{ 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 }};
```

rows \ columns	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

1-D array :-

```
int A[5];
```

$A[0] = *(A+0)$

$A[i] = *(A+i)$

2-D array :-

```
int A[3][4];
```

$A[i][j] \quad *(*(A+i)+j)$

Accessing 2-D array Elements

```
A[i][j] = *(*(A+i)+j)
```

$= *(A[i] + j)$

$= *(A + i)[j]$

Accessing 2-D array Elements

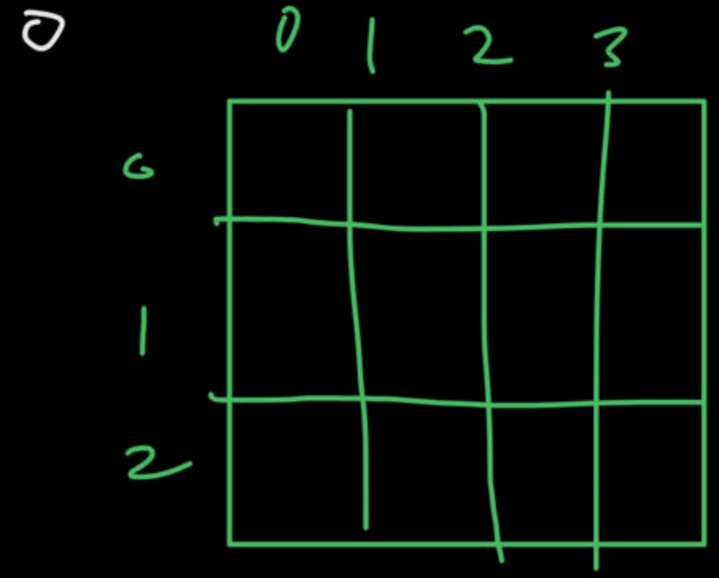
```
int main() {
    int A[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    printf("%d",A[2][3]); '2
    printf("\n%d",*(*(A+1)+3)); 8
}
```

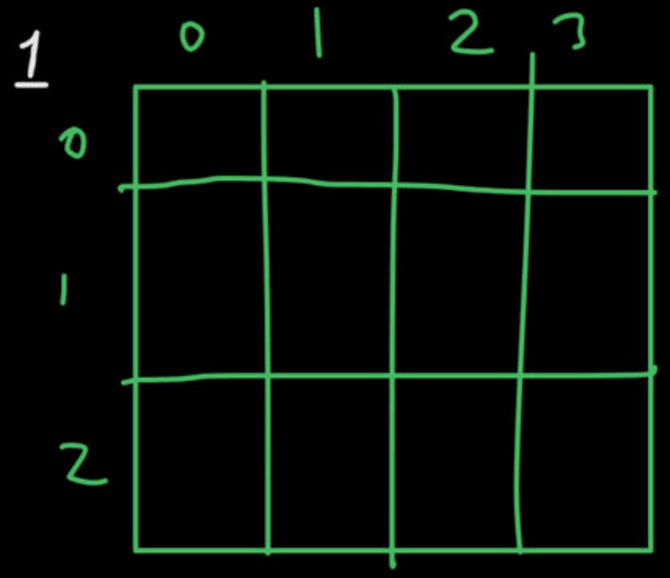
3-D Array

2-d array ⇒ collection of 1-d arrays

3-d array ⇒ —11— 2-d arrays

int A[2][3][4]





A[0][2][3]

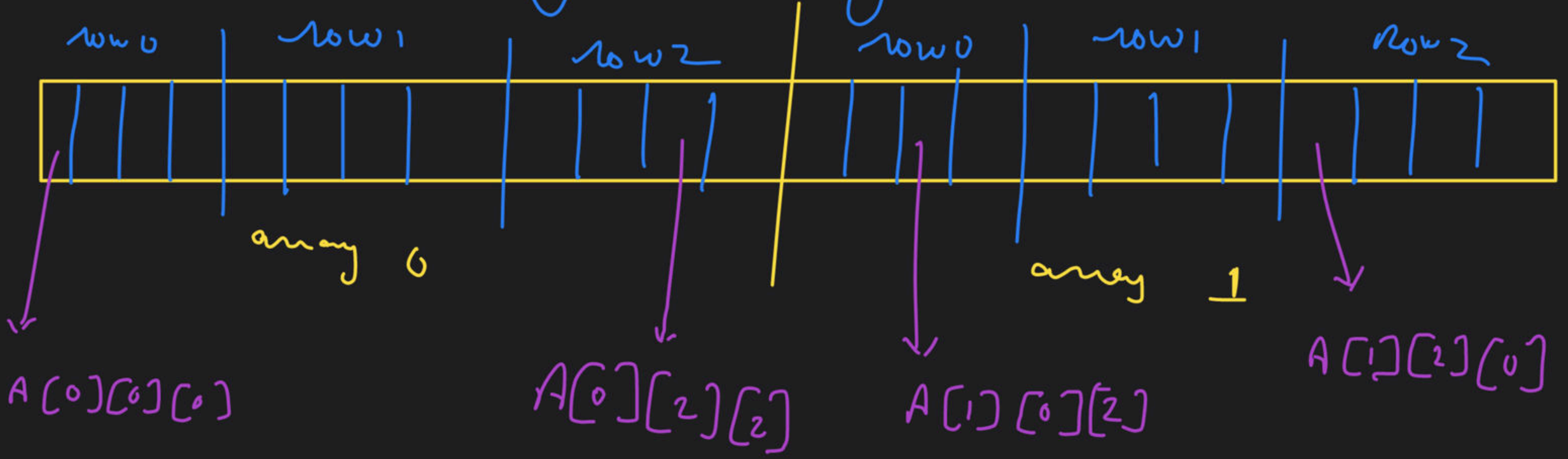
A[i][j][k]

array ⇒ i

row ⇒ j

column ⇒ k

storage of 3-d array in memory



$A[i][j][k] \iff *(*(*(A+i)+j)+k)$

A + 0 ⇒ 0^{th} array

A + 1 ⇒ 1^{st} array

A + 2 ⇒

A[0] + j ⇒ in 0^{th} array row j

A[0][0] + k ⇒ in 0^{th} array and 0^{th} row at column k

Void Pointer

Datatype ⇒ Void ⇒ meaning "no any type"

int *p; pointer to integer

Void *q; pointer to no any type

Void Pointer

```c
int main() {
    int a=5;
    char b='V';
    void *p;
    p=&a;
    printf("%d",*p);
}
```

error → needed

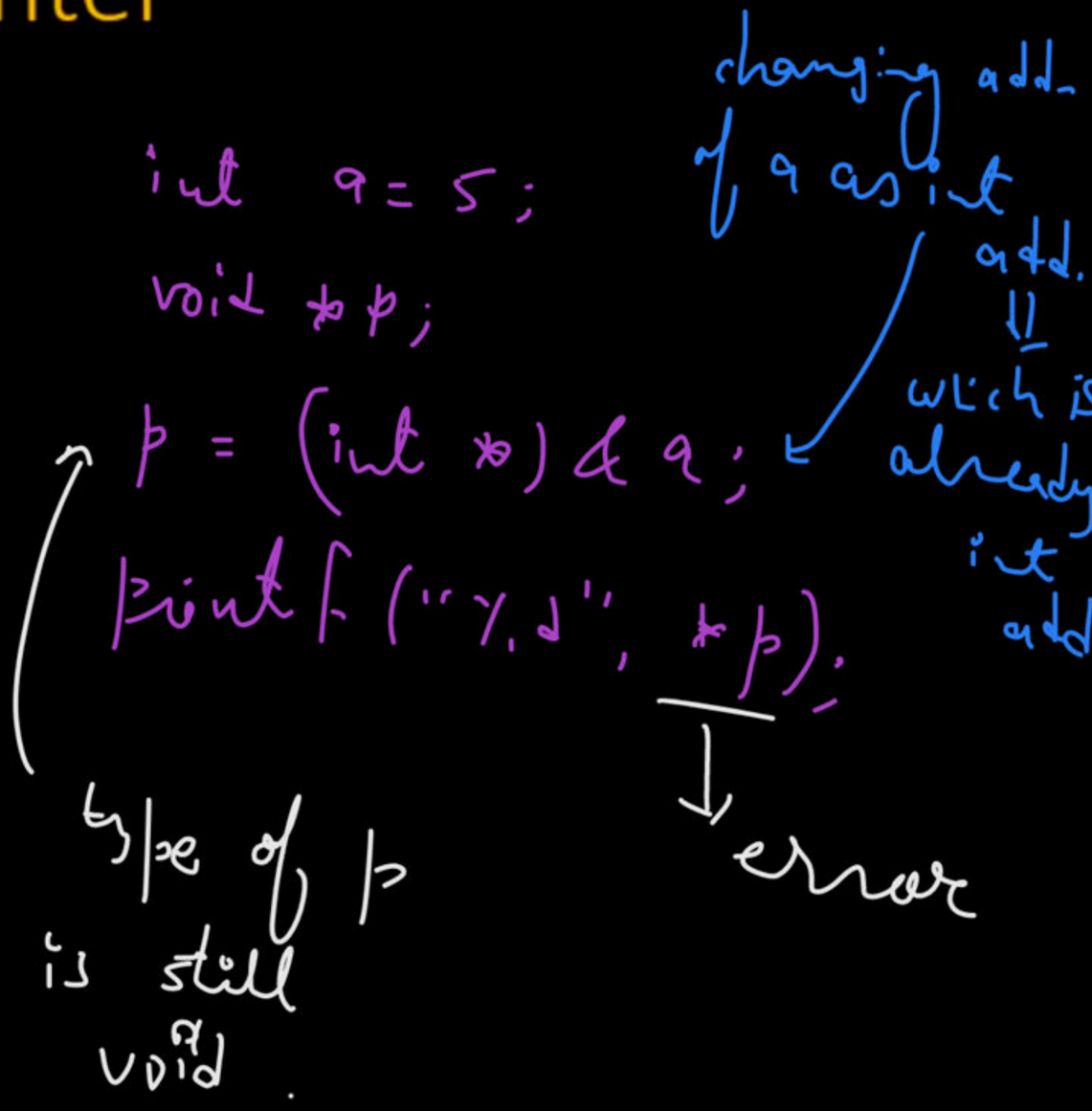
```
*(int*)p;
```

Void Pointer

```c
int main() {
    int a=5;
    char b='V';
    void *p;
    p=&a;
    printf("%d\n",*(int *)p);
    p=&b;
    printf("%c",*(char *)p);
}
```

5

V



Structure

↳ collectn of diff. - datatype elements

Req. =>

Book infon => Book_id => int
Book_Quantity => int
Book_price => float

=> Collectn => structure

Declaration :- outside main function

```
struct name
{
    datatype    name;
    datatype    name;
    .
    .
    .
};
```

ex:

```
struct book
{
    int id;
    int quantity;
    float price;
};
```

← making a user defined datatype

⇓ to use it

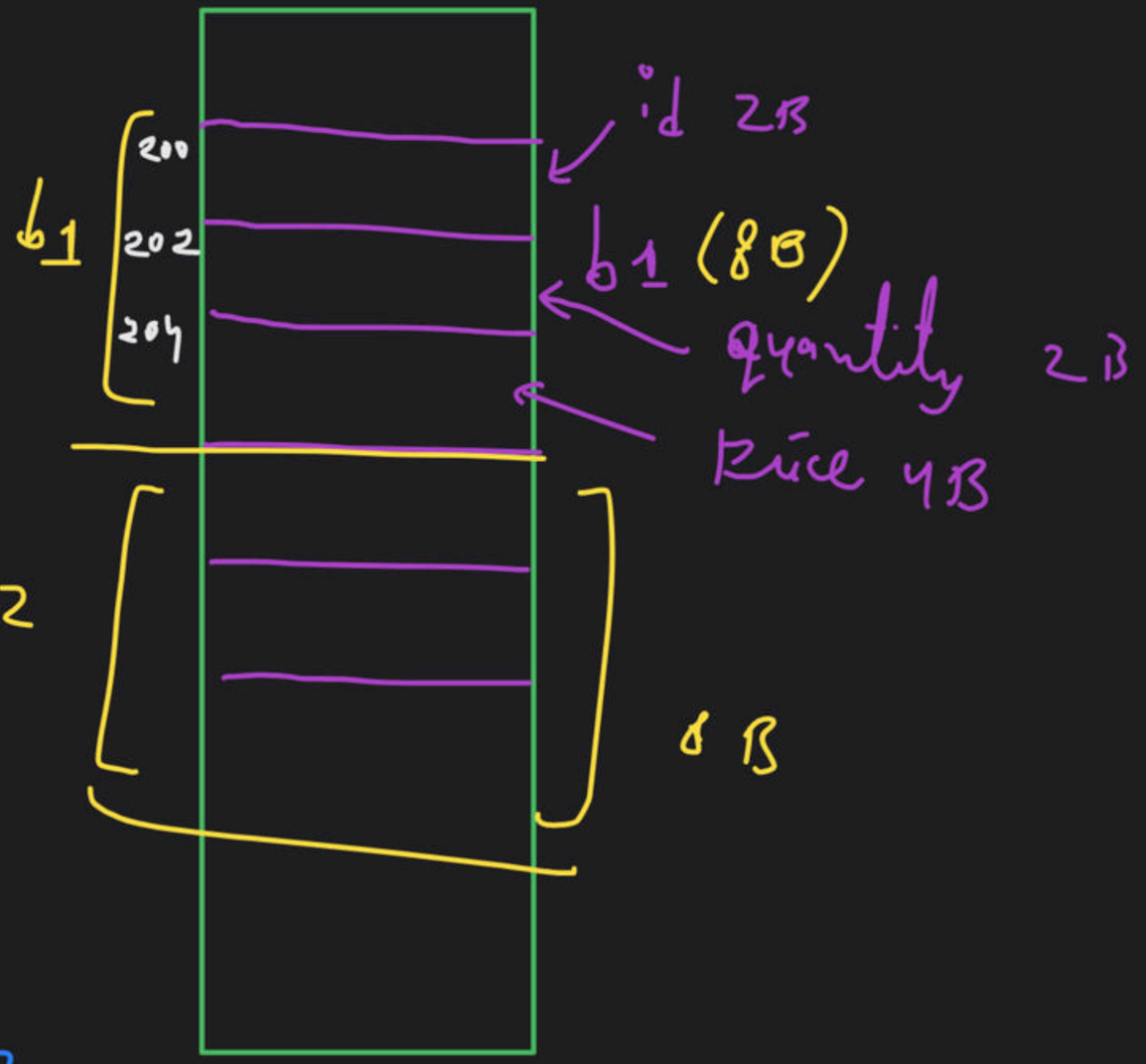
⇓ create variables

```
void main()
{
  struct book b1, b2;
  scanf("%d", &b1.id);

  printf("%d", b1.id);
  printf("%u", &b1.id);
  printf("%d", sizeof(b1));
}
```

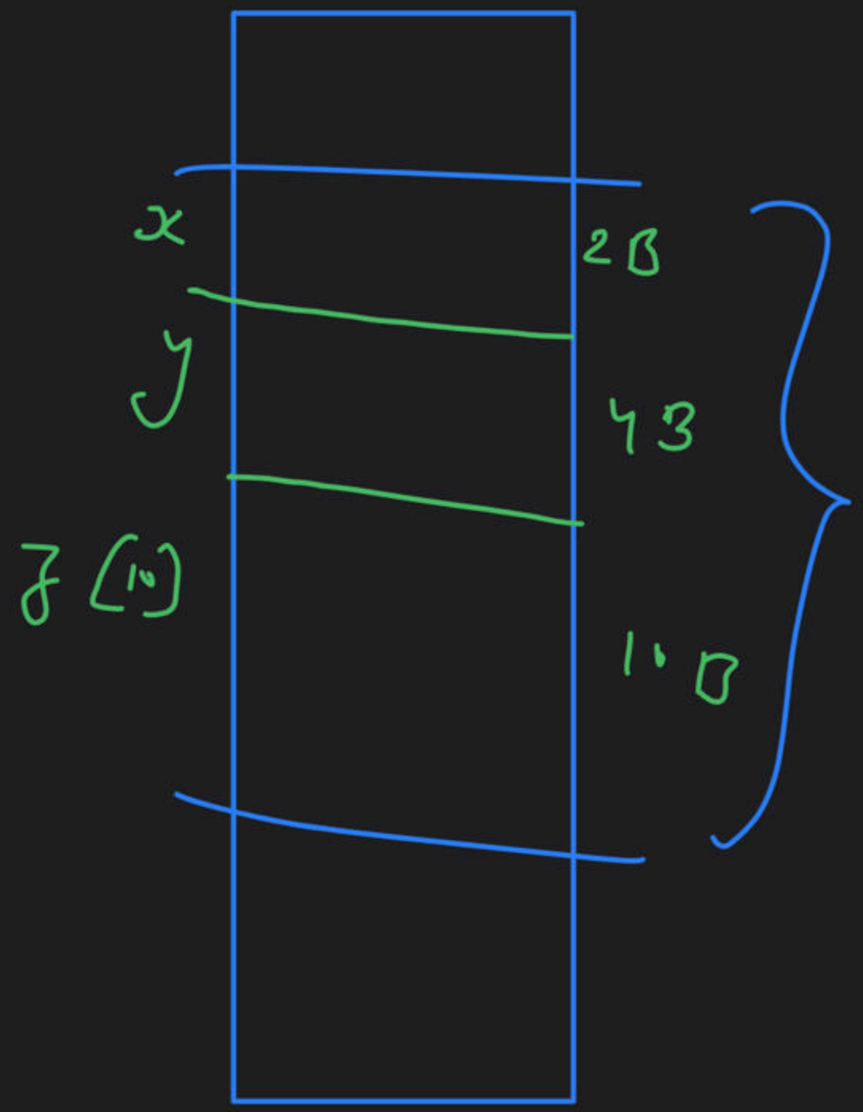
=> 200

8



```
struct test
{
    int x;
    float y;
    char z[10];
};

void main()
{
    struct test t1;
    t1.z[0];
    t1.z[i];
```



printf("%d", sizeof(t1)); 16

```
struct test2
{
    struct test1 t1;
    =
    =
};
struct test2
    abc;
abc.t21.x;
```

t1

```
struct test
{
   int *p;
   char *q;
};
```

```
struct test
{
   int x;
   struct test y;
};
```



```
struct test
{
   int x;
   struct test *y;
};
```

A structure variable can not be the member of own structure

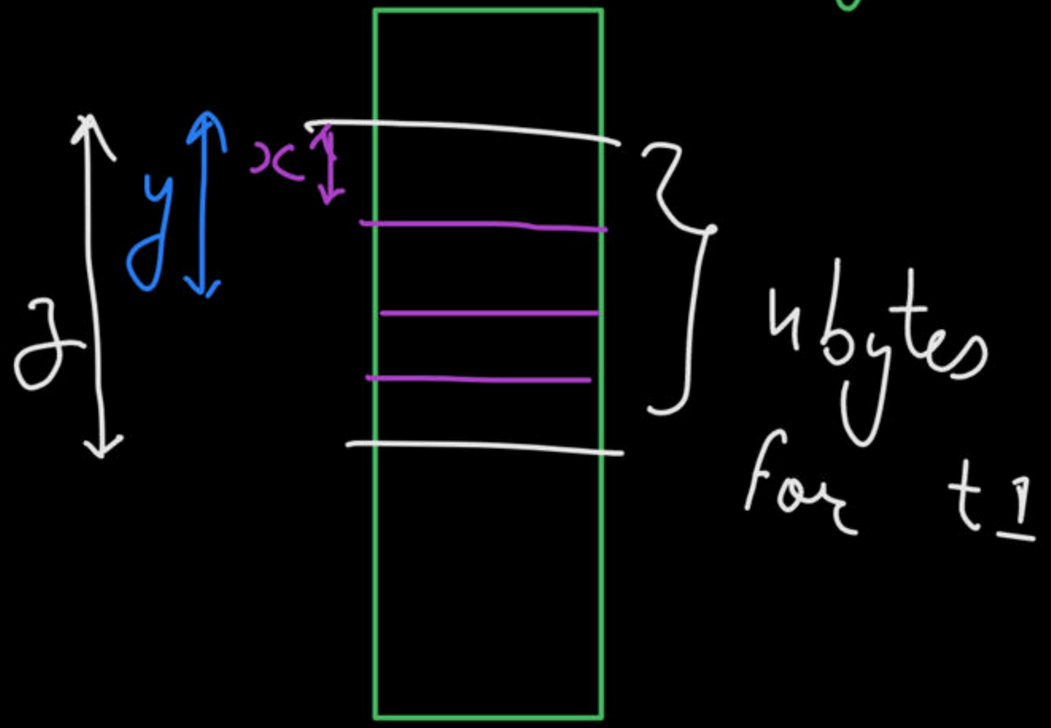
self-referential structure

Union

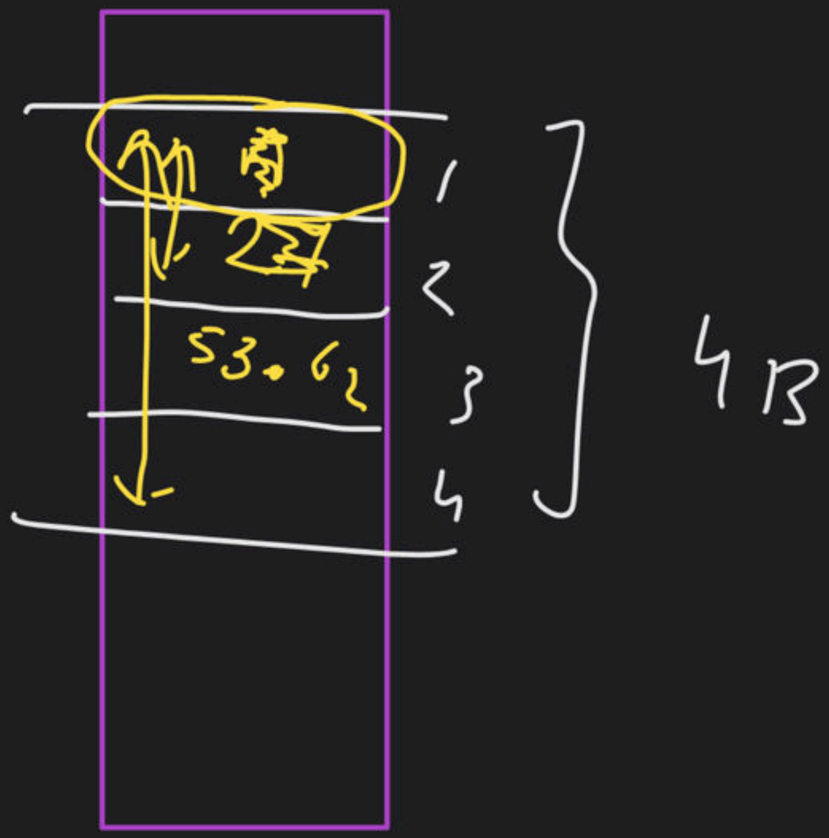
↓

elements stored on overlapping spaces

```
union test
{
    char x;
    int y;
    float z;
};
void main()
{
    union test t1;
```



```
t1.x = 'A';
printf("%c", t1.x);
t1.y = 27;
printf("%d", t1.y);
t1.z = 53.62;
printf("%f", t1.z);
printf("%c %d", t1.x, t1.y);
```



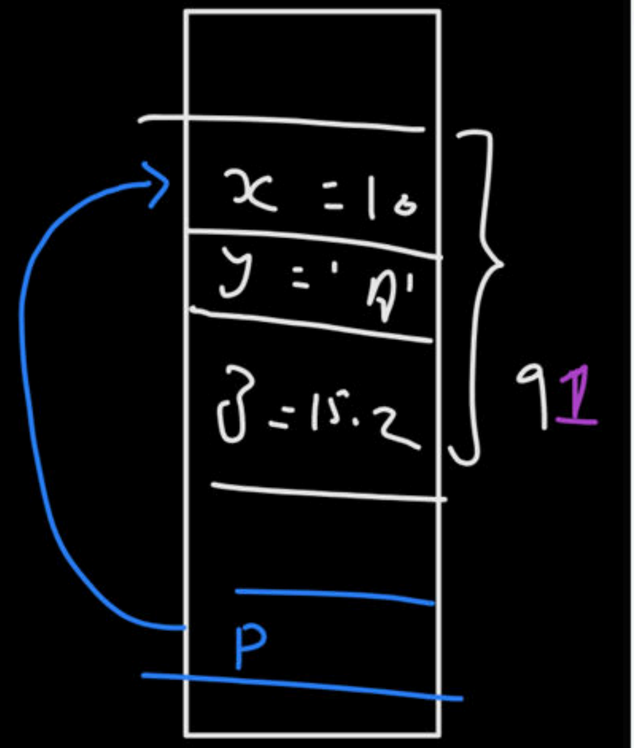
Structure Access Using Pointer

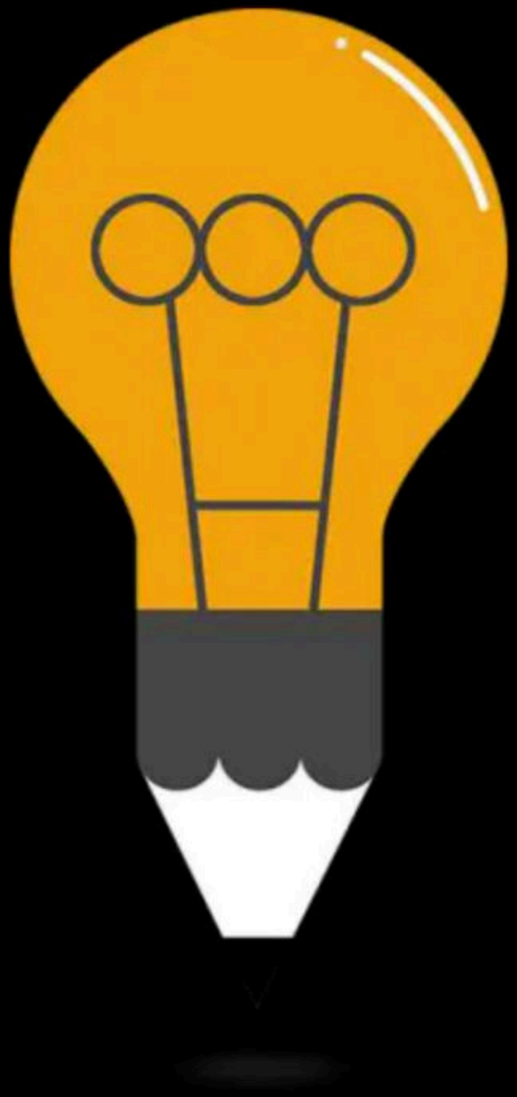
```
struct ABC
{
  int x;
  char y;
  float z;
};
void main()
{ struct ABC a1;
  struct ABC *p;

  a1.x = 10;
  a1.y = 'A';
  a1.z = 15.2;
```

```
p = &a1;
printf("%d", p->x);
```

$p \rightarrow x \iff *p.x$





DPP 4

By: Vishvadeep Gothi

Question

What is the output of the following programs-

```
void main(){
int x[][3]={10,20,30,40,50,60};
int (*y)[3]=x;
printf("%d %d ",(*y)[1],(*y)[2]);
++y;
printf("%d %d",(*y)[1],(*y)[2]);
}
```

Question

What is the output of the following programs-

```
void main(){
char *x[]={“GATE”,”EXAM”,”WORK”,”HARD”};
char **y[]={x+3,x+2,x+1,x};
char ***z=y;
void main(){
printf(“%s”,**++z);
printf(“%s”,*--*++z+3);
}
```

Question

What is the output of the following programs-

```
void main()
{
int x[2][3]={{1,2,3},{4,5,6}};
printf("%d",sizeof(x)/sizeof(int));
printf("%d",sizeof(x[0])/sizeof(int));
printf("%d",sizeof(x[0][2]);
}
```

Question

What is the output of the following programs-

```
void main()
{
int a[][3]={10,20,30,40,50,60,70,80,90};
printf("%d,%d",1[a][2],*1[a]);
}
```

Question

If A is one dimensional array, A[i] is evaluated as

(a) A+i
(b) *A
(c) *(A+i)
(d) *A+i
(e) None of the above

Question

If A is two-dimensional array, A[i][j] is evaluated as

(a) (A+i) + j

(b) (*A+i)+j

(c) (*(A+i)+j)

(d) *(*(A+i)+j)

(e) None of the above

Question

Which of the following is/are valid declarations?

a) int a[2][3]={1,2,3,4,5,6};
b) int a[2][3]={{1,2,3},{4,5,6}};
c) int a[][3]={{1,2,3},{4,5,6},{7,8,9}};
d) int a[2][]={{1,2,3},{4,5,6}};

Happy Learning.!