-: Lecture-8 :-

Programming:- (8-10) marks ✓

1. Scope of a Variable
   - I) static scoping
   - II) dynamic scoping
2. Static Variable, Extern Variable, Auto Variable.
3. Pointers
4. structures, files.
5. Arrays.

1) Scope of a Variable → static scoping. / dynamic scoping

Ex ① Var a, b : Integer // Global Variable.

```
procedure P; // function P
    a=5;
    b=10;
end(P);
procedure Q; // Function Q.
    Var a, b : integer // Local declaration.
    P;
end(Q);
begin. // In old language No name means main program.
    a=1;
    b=2;
    Q;
Write a, b;
end.
```

② ③ ① ④

K/a Free variable.

// Variable is Not declared.

- Without declaring any variable you can't use it. Firstly declared it then memory will be allocated it automatically & then use it

← within fn declaration ✓

★★★ // declaration Time memory will allocate.

★★★ local declaration Memory will allocate at Run Time → memory will allocate in m.m stack portion

★★★ Global declaration memory will allocate at compile time → memory will allocate in m/m Static Portion

Scanned with CamScanner

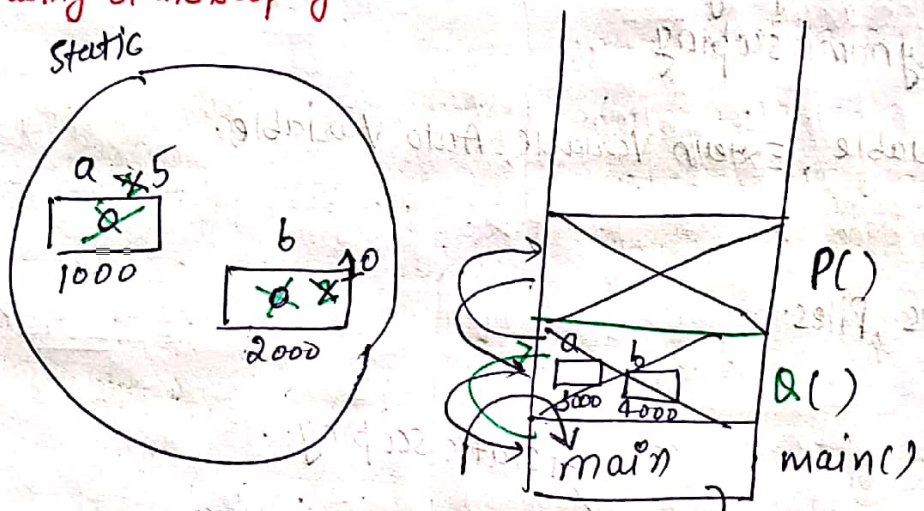
• Whenever a Function is call then it will Push Inside the stack.

• By default Global Variable have 0

By default Local Variable have garbage Value

Previous program solⁿ

① By using static scoping.



Ans (5,10) Ans

By using local Variable.

Write a, b;

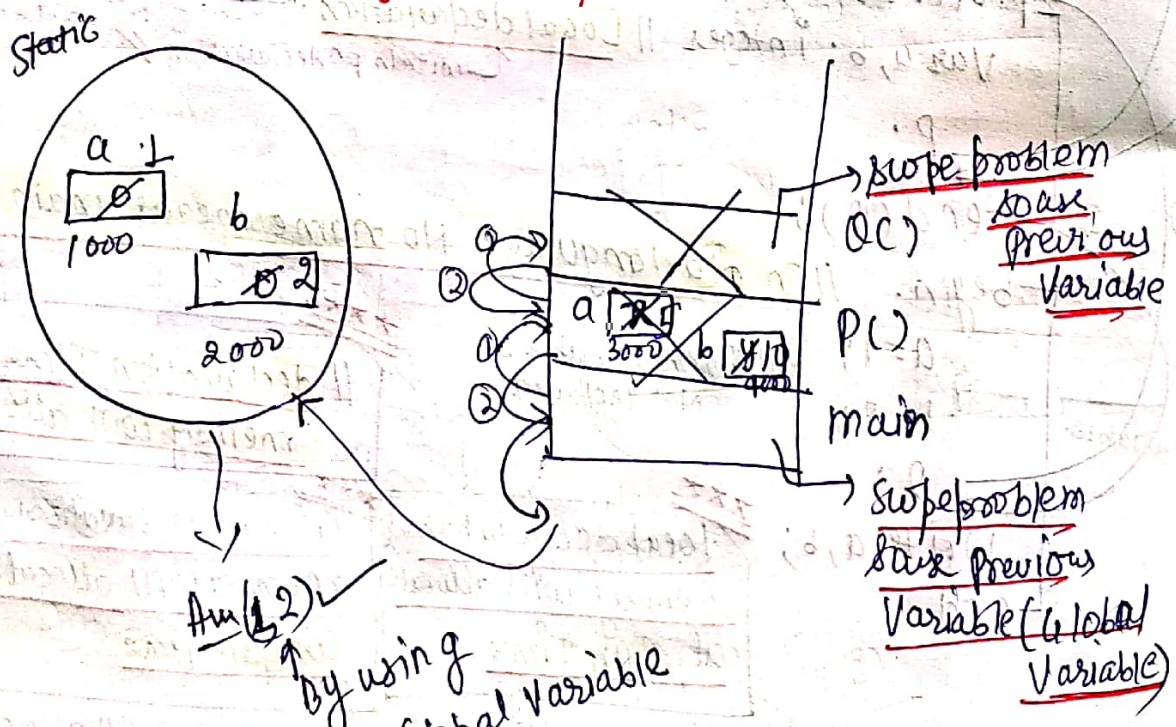
5 10

problem

• If you use Any Variable without declaration K/a scope.

• If their is No Local Varible present then use Global Variable this is K/a static scoping.

② By using dynamic scoping {Previous problem solved}.



Ans (1,2)

By using Global Variable

Static scoping:-

↳ If scoping problem comes then compiler will resolve the problem

Dynamic scoping

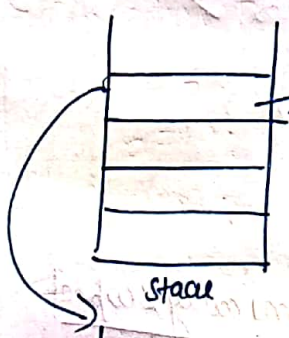
↳ If scoping problem comes then problem will be resolve at run Time. (Processor will Resolve)

- static & dynamic scoping Both solve your problem but static scoping solve your problem easily.

- Static scoping (Easy in Nature)

↳ static scoping says that If any Variable is Not declared & If their exist a Global Variable then Take That global Variable directly

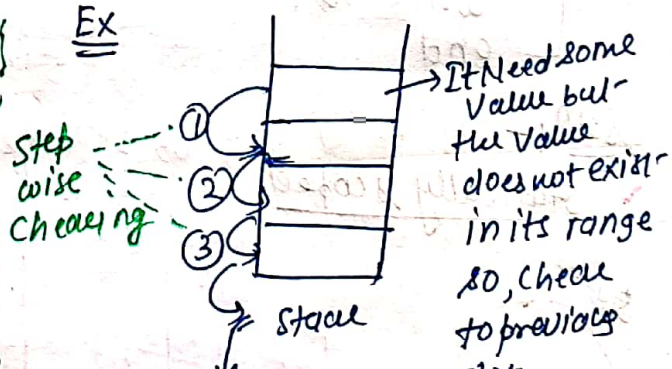
Ex



Dynamic scoping (Very few use in computers)

↳ Dynamic scoping says that if any Variable is Not declared then Take previous Variable value & use if their is No previous Variable value present then use Global Variable value in worst case

Ex



↳ By default we use static scoping

Because dynamic ~~set~~ scoping have so many push & so many pop.

Scanned with CamScanner

Ex2) Var x, y integer;

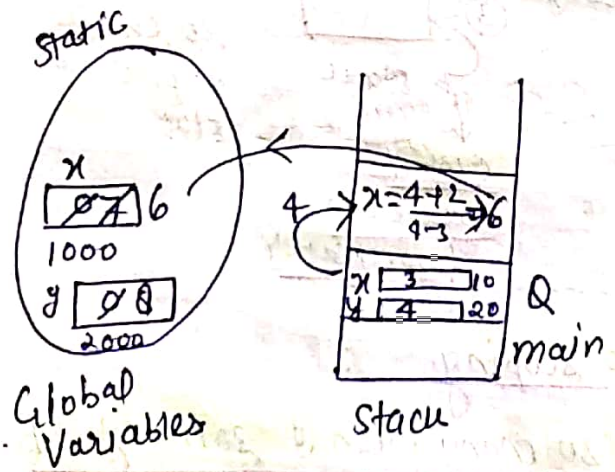
```
Procedure p(n: integer)
  begin
    x = (n+8)(n-3);
  end

procedure Q
  Var x,y : integer;
  begin
    x := 3;
    y := 4;
    p(y);
    write (x);
  end

begin
  x := 7;
  y := 8;
  Q;
  write (x);
end
```

What will be printed by write statements if the variables are statically scoped/ dynamically scoped. ?

(A) 3,6	(B) 6,7	(A) 3,6	(B) 6,7
(C) 3,7	(D) None	(C) 3,7	(D) None

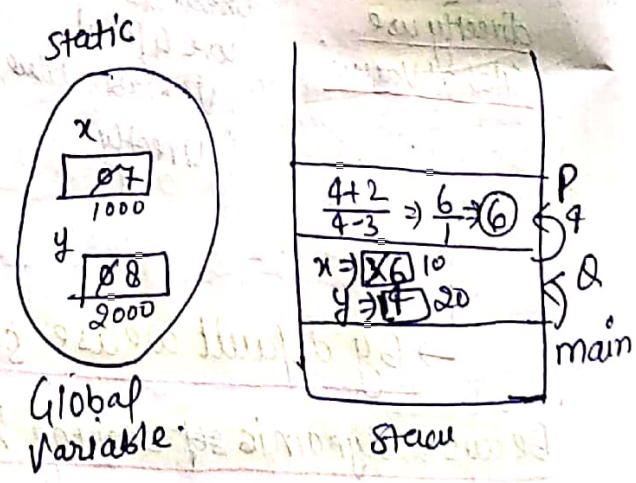
Soln:-

Statically scoped



Ans 3, 6 Ans (a) ✓

dynamically scoped



(6,7) Ans (B)

Scanned with CamScanner

Que find o/p of the program by using static scoping & by using dynamic scoping ✓

★★★ This will be assume as local variable which is declared

```
int x=5, y;
main()
{
  x=3;
  A();
  printf(x,y);
  B(x);
}

A()
{
  printf(y);
  x=7;
  B(y);
  x=8;
  y=9;
}

B(int x)
{
  printf(x,y);
  C(x,y);
  x=11;
  y=36;
}

C(int y, int x)
{
  x=2;
  y=1;
  printf(x,y);
  x=5;
  y=6;
}
```

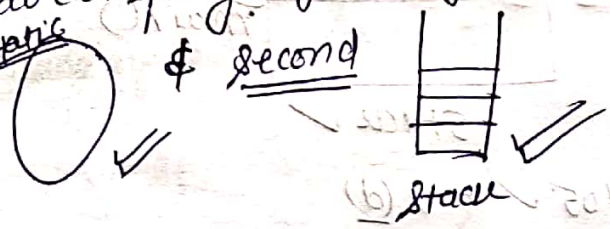
Sol^n :-

① Static scoping ✓

O/p Ans → (000 21 89 8921)

Hint → when we pass variable to F^n (B) & F^n (C) then B & C store ~~in to~~ into it's local variable so, do carefully by taking two memory

First static & second



② Dynamic scoping ✓

O/p Ans → (000 21 89 8921)

O/p is same but ~~step~~ approach is different. If variable Not found in it Block then check Previous one {dynamic scoping}

But In static scoping → If variable Not found in it's Block then check global variable directly ✓

Scanned with CamScanner

Que Consider the following program —

```
global int i=100, J=5;
Void P(x)
{ int i=10;
  Print(x+10);
  i=200;
  J=20;
  Print(x);
}
main()
{
  P(i+J);
}
```

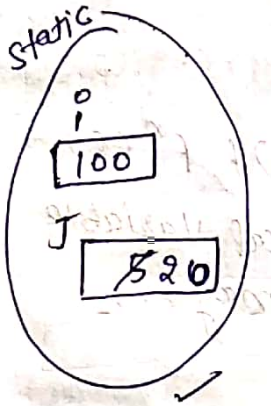
→ this will be treated as local Variable

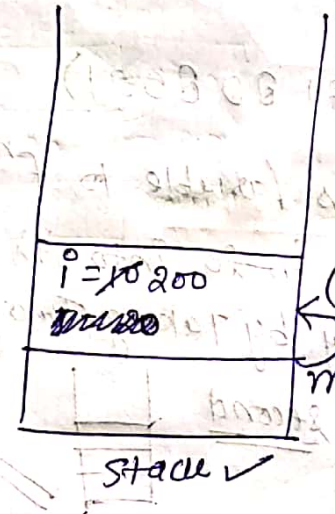
If the above programming language uses static scoping then the Values printed by the above program are:-

(A) 115, 220 (B) 25, 220

(C) 25, 15 (D) 115, 105

Sol^n





O/P ⇒ 115, 105 Ans (d)

Que What will be the output of the following Pseudo-code using dynamic scoping ?

```
a=3;
Void n(x)
{
  x= x*a;
  Print(x);
}
Void main()
{
  m(a);
}
```

Here at both place only initialization occure But this type of Initialization if occure at both place then it is also a declaration

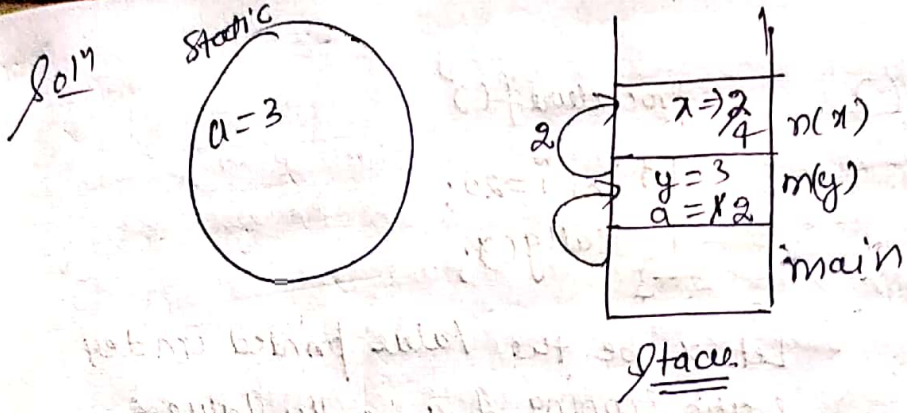
```
Void m(y)
{
  a=1; // Local declaration
  a= y-a;
  n(a);
  Print(a);
}
```

(a) 6,2 (b) 6,6

(c) 4,2 (d) 4,4

Scanned with CamScanner

Sol^n



Ans (C) 4, 2 ✓

* If we do this program by static scoping then our answer will be → (6, 2)

- Initialization (No memory allocate) → declaration (memory allocation)
  - (a=1)
  - (int a)

- Modern language all follow static scoping.

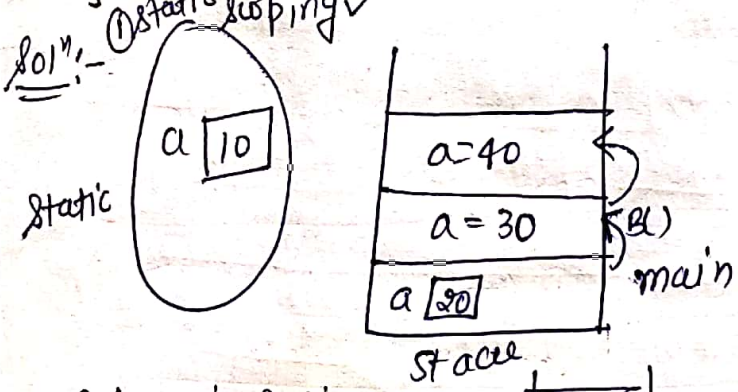
Que What is the o/p of the program —
a) By using static scoping
b) By using dynamic scoping

```
int a=10;
main()
{
  int a=20;
  B();
}
```

```
B()
{
  int a=30;
  C();
}
```

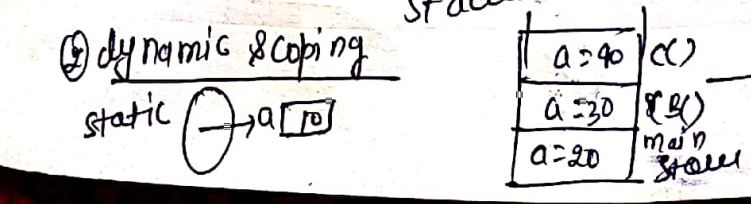
```
C()
{
  int a=40;
  printf(a);
}.
```

Sol^n:- (1) static scoping ✓



→ Ans 40 ✓

(2) dynamic scoping



→ Ans 40

Scanned with CamScanner

Que int i

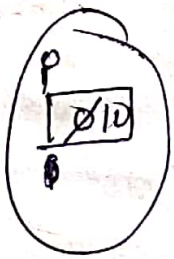
```
Program main()
{
  i = 10;
  call f();
}

procedure f()
{ int i = 20;
  call g();
}

procedure g()
{
  print i;
}
```

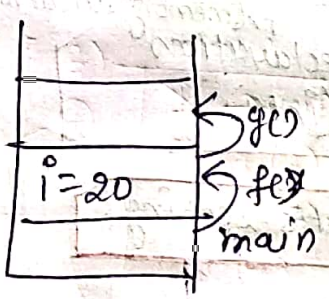
Let x be the value printed under static scoping & y be the value printed under dynamic scoping. Then x & y are —

(A) x=10, y=20 (B) x=20, y=10

(C) x=10, y=10 (D) x=20, y=20

Sol^n:- Static





$\boxed{x=10,\ y=20}$ Ans (A) ✓

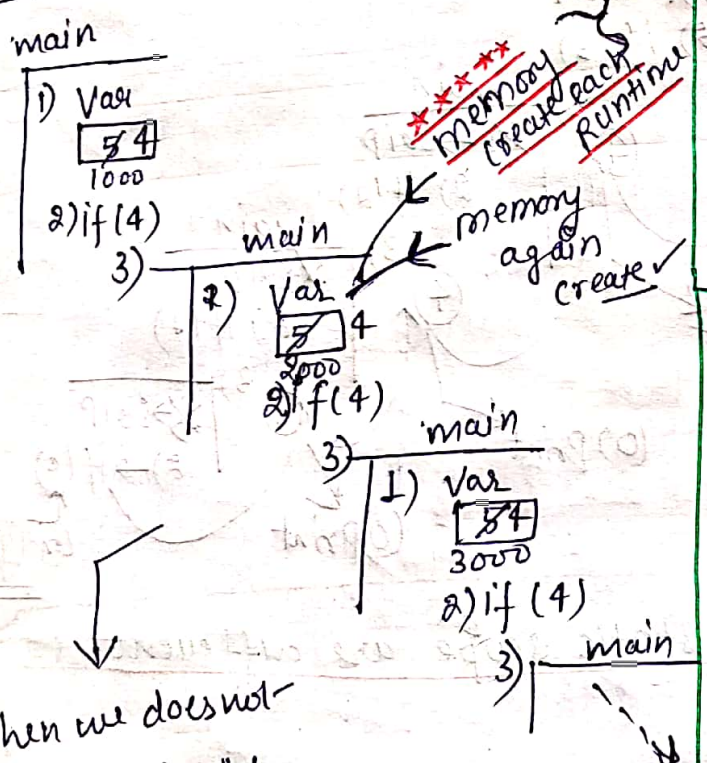
-: Lecture - 9 :-

Static Variable :-

Ex ①

```
main()
{
    static int Var=5;
    if (--Var)
    {
        main();
        printf("%d", Var);
    }
}
```

Sol^n ① Without static



When we does not put static in the variable then It is go to Loop

↳ (Stack over flow Error message get)

Compile Time You don't get any Error

Error get at runtime

Concept of pre-Increment & Post-Increment.

if x, y = 5

$\boxed{x=y++}$ → Tells that Increment the value of y after assigning it to x. It is also K/a Post Increment

So x=y++

Print (x, y) ⇒ 5, 6.

↳ First assign then Increment } Post Increment

$\boxed{x=++y}$ → First Increment then assign.

So $\boxed{x=y=6}$

↳ also K/a Pre Incrementation.

What is o/P → conditional stmt is true always for Every Non zero.

```
if (-210) → +ve or -ve Any Number
Pf(hi)
else
Pf(Byee)
```

Here then hi will be Printing

⇓

if rather than +ve or -ve only for "0" Byee will be printing

***

• In memory where program is running there are three type of memery —

① Static Area → Compile time memory allocate

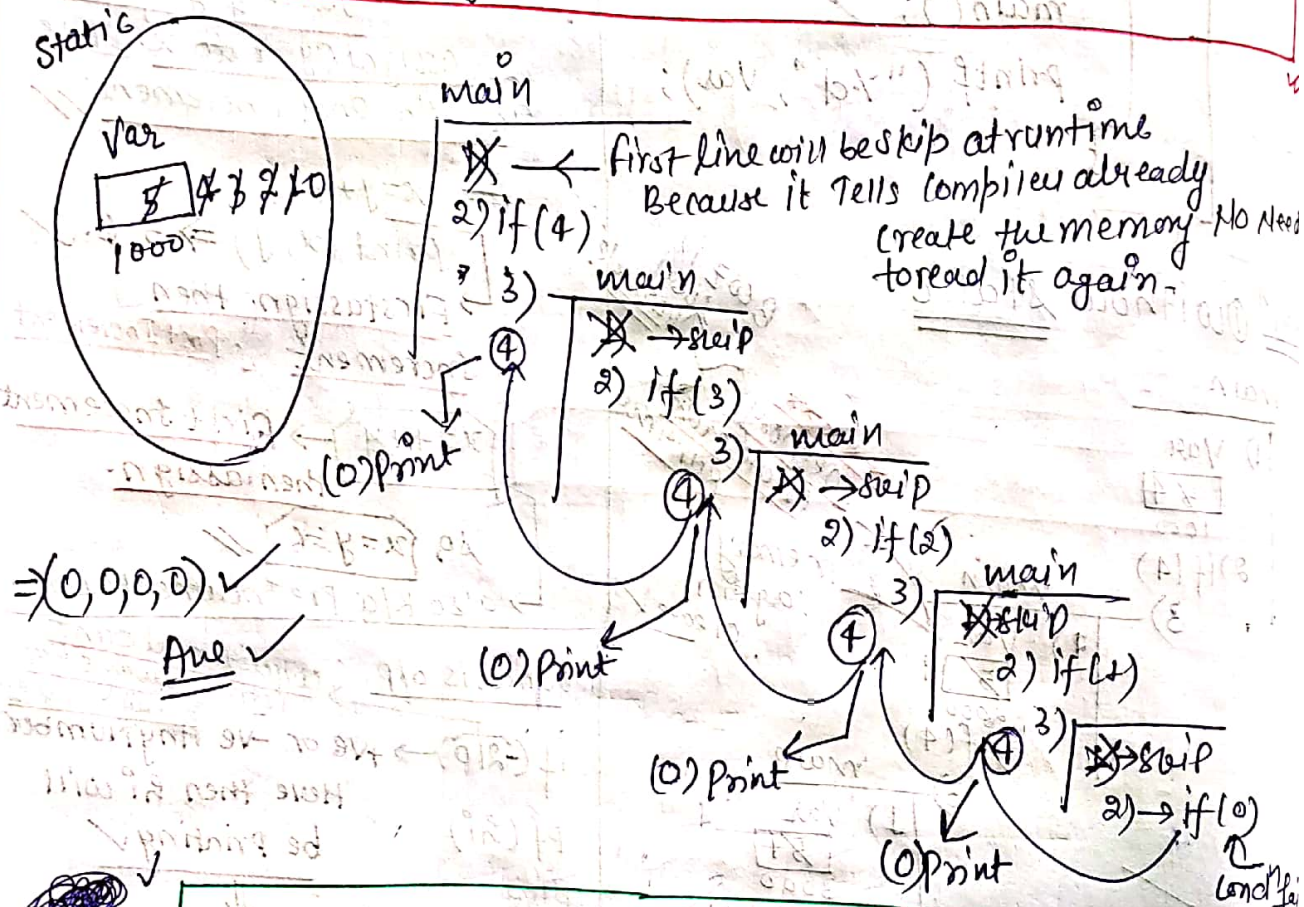
② Stack Area } Runtime memory allocate

③ Heap Area → Runtime dynamically memory allocate

recursive program ran here

② with static o/p of previous program.

- Local Variable memory ~~will~~ will allocate in stack area ✓
- Simply at Local Variable memory will allocate in runtime but the Variable which are static in Nature like (static int x=5) will allocate memory to static area at Compile time only.
- Global Variable memory will ~~allo~~ allocate in static portion at compile time only



Static Variable & static scope are different to each other. ✓

For static Variable

① For static Variable memory will allocate Compiler time by Compiler in static area of memory only once ✓

② For static memory initialization will be done only once

③ By default static Variable will be initialized to zero.

↳ B Coz of storing into global memory ✓

• By default static Variable will be Initialized to ⇒ 0
↳ All activity working like Global Variable.

· Stack overflow Error message occur at run Time.

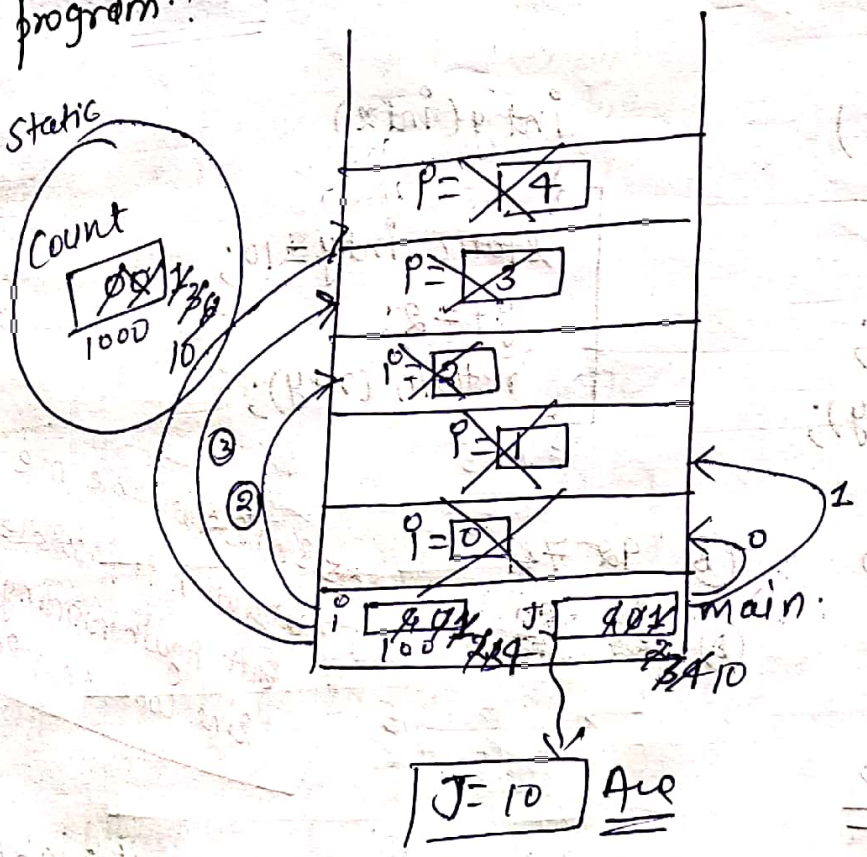
Ex:-

```
main()
{
    int i,J;
    for (i=0; i<4; i++)
        J=incr(i);
}
```

```
int incr (int i)
{
    static int count=0;
    count = count+i;
    return (count);
}
```

The Value of J at the end of the above program?

Sol^n:-



Ans

* Count memory created & Initialised only one Time but It is updated when ever i is Incremented in incr(int i) function call.

Que What is the Output of the following program—

```
#include <stdio.h>
int f (int x);
int g (int x);
main ()
{
    int x=5, y=10, counter ;
    for (counter =1 ; counter <=2; ++counter)
    {
        y += f(x) + g(x);
        printf ("%d", y);
    }
}
```

} → Prototype ✓

```
int f (int x)
{
    int y;
    y=g(x);
    return (y);
}
```

```
int g (int x)
{
    static int y=10;
    y+=2;
    return (x+y);
}
```

(A) 43 80

(B) 42 74

(C) 33 37

(D) ~~32 32~~ 46 90

** Static Variable me updation is possible but Again running of that line code is not Possible

Ans (d) 46, 90

Hint:- Here Value of y in f^n g is Static Variable so, Memory of y will be created statically {which is initialised ~~by~~ at compile Time}. and after Initialization the line static int g=10; will not Run any more time only value of y will be updated ✓

Scanned with CamScanner

Que:- Conside the following C code segment.?-

```
int a,b,c=0;
void f();
main()
{ static int a=1;   /* Line 1 */
  f();
  a+=1;
  f();
  printf("\n %d %d", a,b);
}
```

***this is local for this only main so, Eventhough memory created in static portion but it will be access only by main

```
f()
{ static int a=2; /*Line 2*/
  int b=1;
  a+=++b;
  Pf("\n%d %d",a,b);
}
```

→ same Rule for this

what o/p will be generated. the given code segment

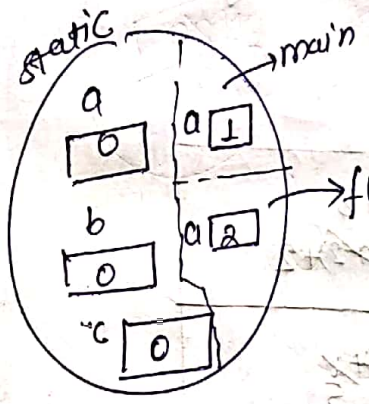
A) 3,1 / 4,1 / 4,2

(B) 4,2 / 6,1 / 6,1

(C) 4,2 / 6,2 / 2,0

(D) 3,1 / 5,2 / 5,2

Sol^n



Ans (C)

4, 2
6, 2
2, 0

What o/p will be generated by the give above program.

if—

Line-1 is replaced by `auto int a=1;`

Line-2 is " " " `register int a=2;`

a) 3,1 / 4,1 / 4,2

(b) 4,2 / 6,1 / 6,1

(c) 4,2 / 6,2 / 2,0

(d) 4,2 / 4,2 / 2,0

Hint⇒ `int a=1 ≡ auto int a=1`

`int a=2 ≡ register int a=2` ⇒ register & int Both are same But when we put register Before int then memory will be created into Register.

auto → means automatically stack portion me

otherwise memory will be created into stack.

Ans (d)

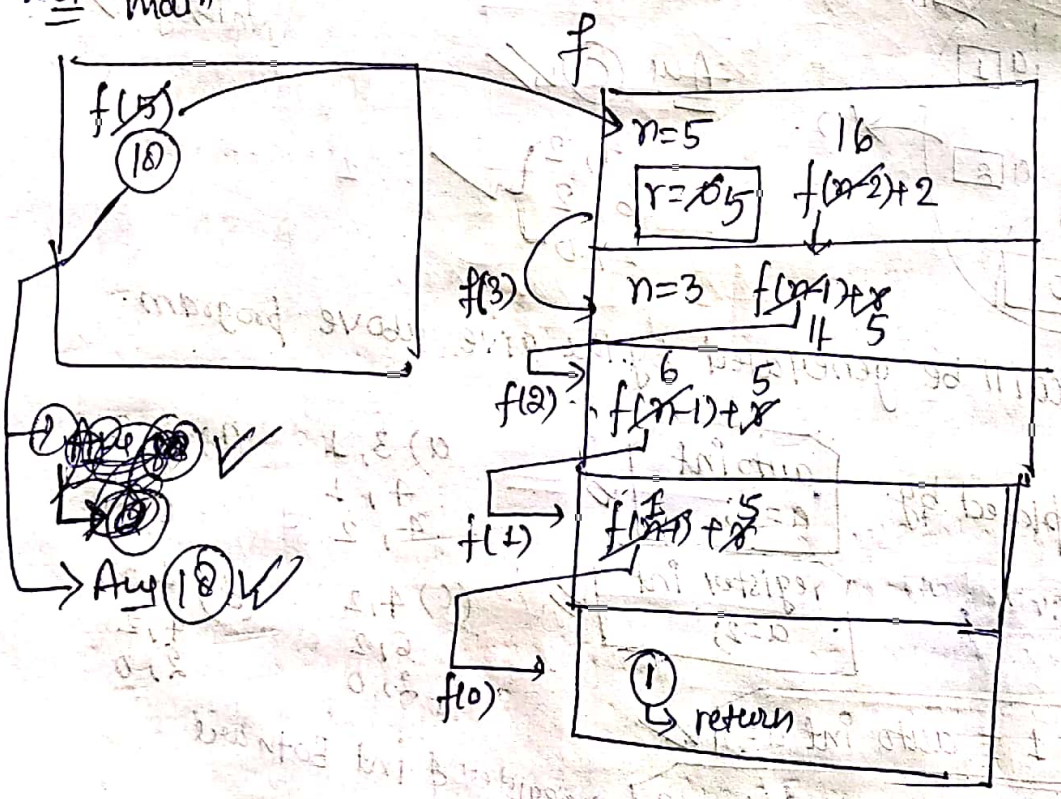
Scanned with CamScanner

auto int = register int = Normal int

Que Consider the following C-function.

```
int f(int n)
{
    static int r=0;
    if (n<=0) return 1;
    if (n>3) {r=n;
              return f(n-2)+2;}
    return f(n-1)+r;
}

main()
{
    printf("%d", f(5));
}
```

Solⁿ main



Ans 18

Que :-

```
int x=10;
main()
{
  int x=1;
  x+= f1()+f2()+f3()+f2();
  printf("%d", x);
  return(0);
}
```

```
int f1()
{
  int x=25;
  x++;
  return(x);
}
```

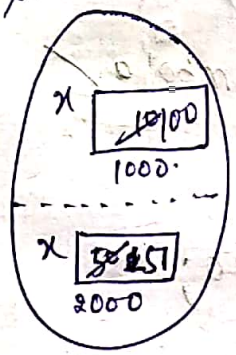
```
int f2()
{
  static int x=50 // Localstatic
  x++;
  return(x);
}
```

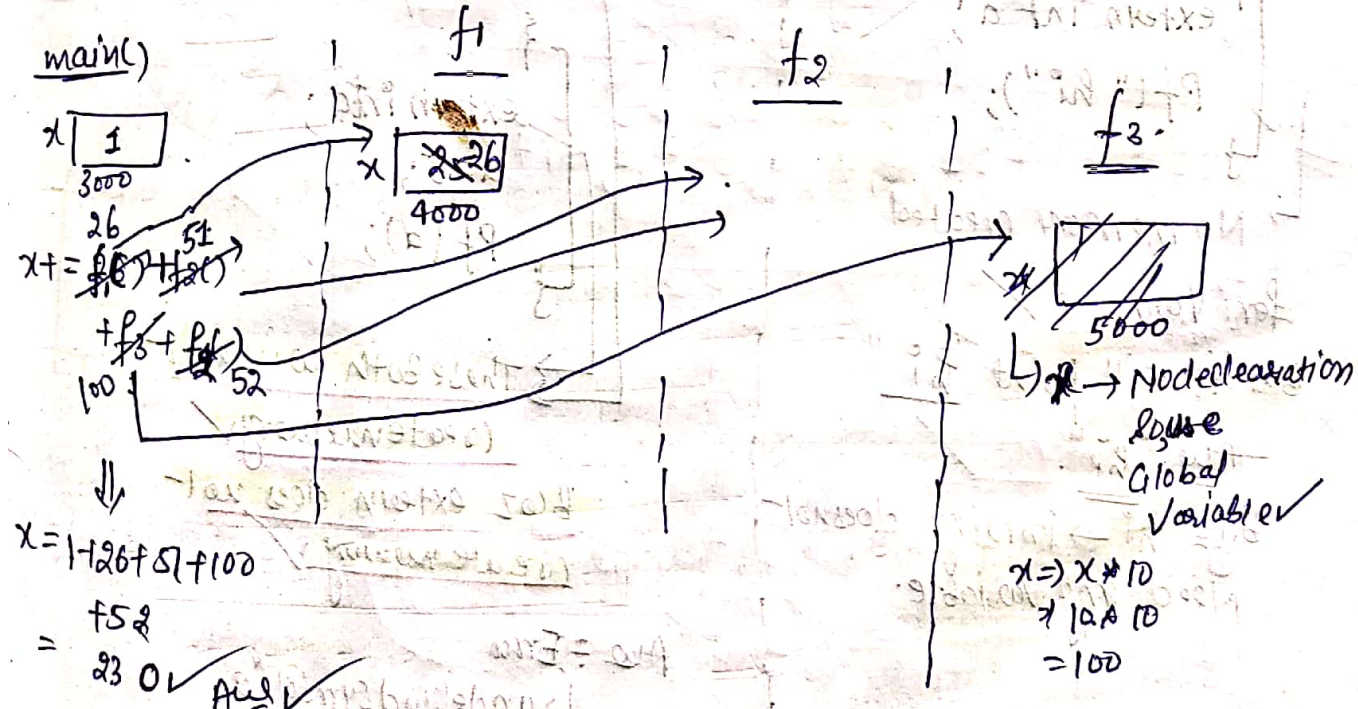
Static ka memory Alayse globally Bnega ✓

```
int f3()
{
  x*= 10;
  return(x);
}
```

What is the output of This program —

Sol^n Static





⇓

x = 1+26+51+100
+52
= 230 ✓ Ans ✓

Scanned with CamScanner

Extern Variable :-

→ extern int a :- When program runs this line says ~~m~~ I am a local variable & my name is a but I want to use externally create memory to use Because I don't want to waste memory. Then processor allocate it to extern memory

Que

```
int a=25;
main()
{
  extern int a;
  a=10
  Pf(a);
  a=30;
  Pf(a);
}
```

Local declaration No memory will created ✓

It use Global memory

a [~~25~~ ~~10~~ 30]
1000

Ans 10, 30 ✓

Que

```
main()
{ extern int a;
  Pf(a);
}
```

extern does not create memory it uses Outside (global) memory only.

But Here No Global memory created.

So, Overall No memory created ✓

Ans Error
↳ undefined symbol a ✓

Que

```
main()
{
  extern int a;
  Pf("hi");
}
```

↳ No memory created for variable ✓

But it will print "hi"

Ans "hi".

Bcoz hi → Printing does not Need any variable.

Que

```
extern int a;
main()
{
  extern int a;
  a=5;
  Pf(a);
}
```

These Both will not create memory ✓

Bcoz extern does not create memory ✓

Ans : Error
↳ undefined symbol "a"

Scanned with CamScanner

Que

```
extern int a=5;
main()
{
  extern int a;
  a=10;
  Pf(a);
}
```

Sol^n:- Here memory will be created for extern because In this problem Initializsation will be occure of variable (a)

a [~~5~~ 10] 1000

& memory will be created globally (static area).

Ans (10)

Que:-

```
int a=5;
main()
{
  int a=10;
  int a=15;
  Pf(a);
}
```

a [5] 3000

a [10] 2000

a [15] 1000

Sol^n

↳ Error:- multiple declaration for variable (a)

Que

```
main()
{
  extern int a=10;
  Pf(a);
}
```

But Here memory will not be created {Bcoz Extern local Variable can't Initialized}

Que

```
int a=5;
main()
{
  extern int a;
  extern int a;
  extern int a;
  extern int a;
  a=10
  Pf(a);
}
```

a [5] 1000

all extern Variable Point to Variable "a".

Sol^n

a [~~5~~ 10] 1000

Ans 10

*** Biggest advantage of extern keyword is — we can take multiple local variable of same name by using extern keyword & their is No. error message will show.

Scanned with CamScanner

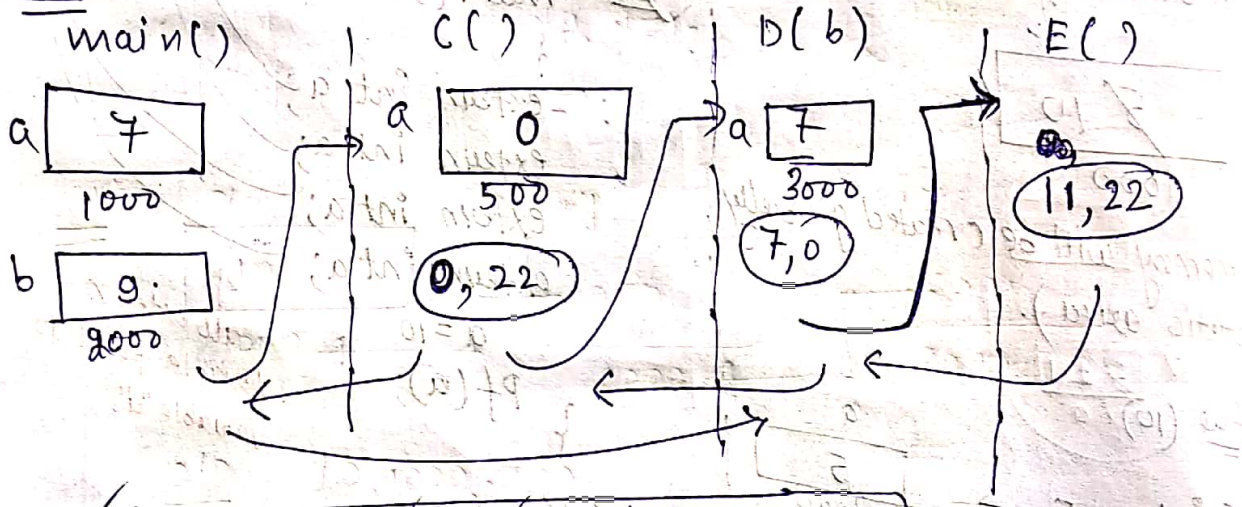
Que What will be the o/p of this program?

```
int a=11, b=22;
main()
{
    int a=7;
    auto int b=9;
    C();
    Pf(a,b);
    D(b);
    Pf(a,b);
}

C()
{
    static int a;
    Pf(a,b);
    D(a);
    Pf(a,b);
}

D(int b)
{
    register int a=7;
    Pf(a,b);
    E();
}

E()
{
    Extern int a;
    Pf(a,b);
    a=4;
    b=7;
}
```

① use static scoping.

Sol^n ①

a [~~14~~] 100 b [~~27~~] 200



O/P 0, 22, 7, 0, 11, 22, 0, 7, 7, 9, 7, 9, 4, 7, 7, 9

zero

Ans

static scoping

** Dynamic scoping

0, 9, 7, 0, 11, 0, 0, 9, 7, 9, 7, 9, 4, 9, 7, 9

Scanned with CamScanner

Summary
* Static Variable :-
* Extern Variable :-
* register " :-
* auto " :-
* static scoping :-
* dynamic scoping :-

-: Pointers :-

int *y → It means that inside y we want to store integer type of Address.

means in y Address store

Ex ① main ()

```
{
int x = 10;
int *y;
printf ("%d %u", x, &x);
y = &x;
printf ("%u %u %d", &y, y, *y);
}
```

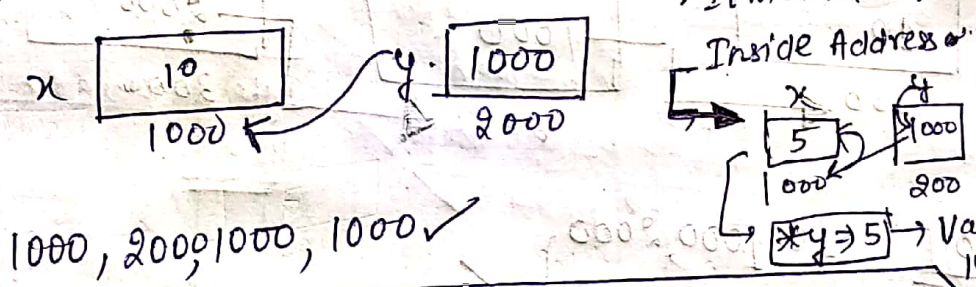
→ // declaration is for O.S that y contain add. of Int typed

→ Here Assume Address size is 8B

[Already given in the Question.

→ *** It means the value Inside Address of y

Sol^n



O/P → 10, 1000, 2000, 1000, 1000 ✓

• Any Variable which Contain Address k/a Pointer Variable

• If you are not assign any address to pointer Variable then that Pointer Variable Contain Garbage Value. → By default Pointer Variable Contain Garbage Value

• In Question it is given that Address size Howmuch then that much size pointer Variable will be created.

from above program ✓

x ← is a integer data Variable.

*y ← is a integer pointer Variable.

Scanned with CamScanner

If I said → I am a character pointer.

It means firstly I am a pointer means I contain Address & if I go to that Address then ~~th~~ then I found character will be stored their.

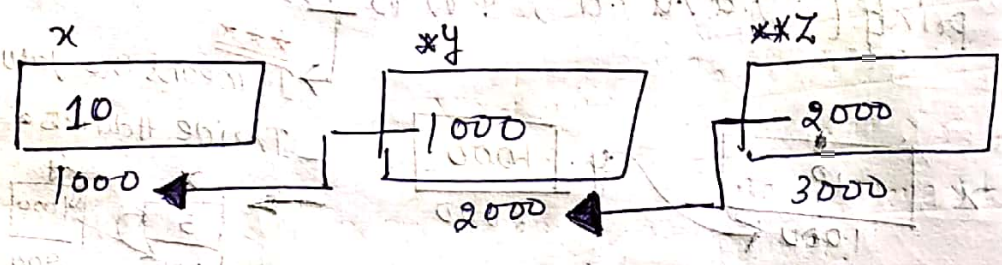
- Address is unsigned Integer so, for Address → %u ✓ use ✓

Ex

```
main()
{ int x = 10;
  int *y = &x;
  int **z = &y;
  printf ("%d %u %u", x, y, z)
  printf (" %d %u", *y, *z);
  printf ("%d ", **z);
}
```

→ Here x → directly data ✓
y → single pointer ✓
z → double pointer ✓

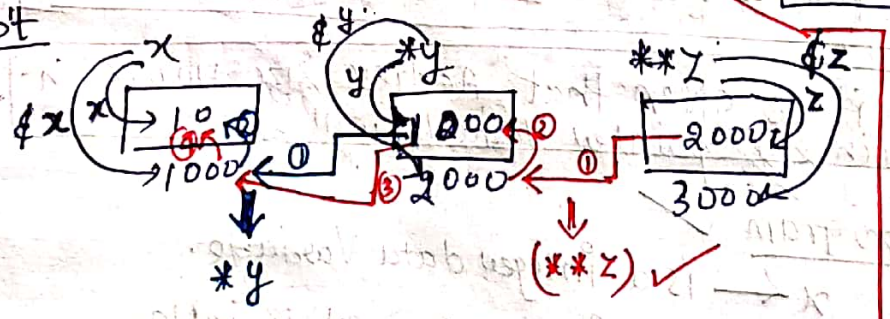
Sol^n



O/P 10, 1000, 2000
10, 1000
10

→ **z → means *(* 2000)
↓ value at 2000 add.
(* 1000)
→ value at 1000 add
→ Ans 10

Concept



• int ** z = &y → Means:- z is a double pointer which contain Address of y.

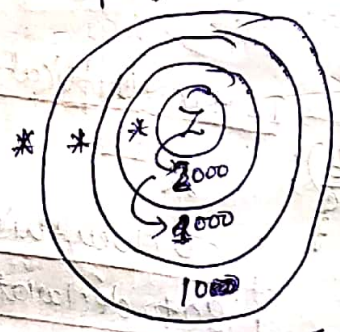
✓ To the single pointer if you give (**) ← double stars (double pointer) to access the data then that will give you an error message. ✓

Ex (*y) ← from previous one.

↳ If you provide (**y) ← called double point

↳ give error message in previous program.

→ from previous program



Error ✓

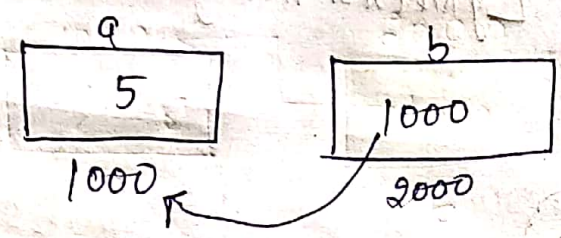
Concept		
① int a=10; int *b =20; ⇓ Here in second line you said b store address but you store data ✓ which is wrong ✓ so error message come. ✱ Some-time error Message does not come because of data size == Add size but if you Initialized Pointer then you should ~~stor~~ store address	② int a= 10; int b = &a; ↓ Here in second line you ~~sho~~ tells b is storing data by using int b, but you store address ✓ which gives error message. Error will Not come when data size == Address size.	③ int a =10; int *b= &a; ↓ ✱ Every ~~this~~ things is perfectly fine ✓ ① Here In 1st line ✓ you said I want to store data & you store it ✓ good ② In 2nd line. You said I want to store Address & you store it ✓ good ✓

Scanned with CamScanner

***Ex

```
main()
{
    int a=5;
    int *b= &a;
    Printf("%d", a*b);
    printf("%d", a**b);
}
```

Soln



from line Printf("%d", a*b);

Is it multiplication (or) pointer.

To check this go to declaration first. In the declaration it is given that *b is a pointer type so Here *b will be pointer.

5 5

No operator in between so error message will be printed ✓

from line printf("%d", a**b);

check from declaration, declaration says that only one is with b so I am Not a Pointer

check from declaration that the * which is with b is a pointer or multiplication.

from declaration it says that I am a pointer

operator

5*(5) → *b

25 Ans

So, O/P ⇒ 25 ✓

Scanned with CamScanner

Que

```
int main()
{
    int a=5, b=6;
    int *p=&a, **q;
    *p=20, q=&p;
    f(a,&b);
    *q=&b;
    *p=30;
    printf("%d %d", a,b);
}

Void f(int x, int *p)
{
    *p=x;
    x=10;
}
```

Whats the output produce by above C-program?

(a) 10, 20 (b) 20, 30

(c) 30, 10 (d) 20, 20

*** → like local variable

· By default pointer contain Garbage value

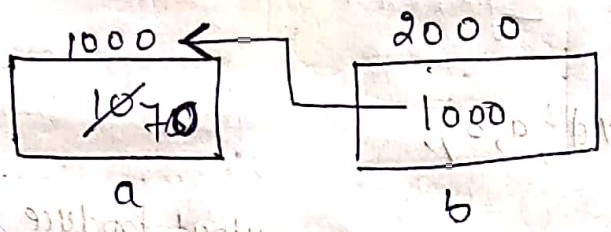
Scanned with CamScanner

Que main()

```
{
    int a=10;
    int *b = &a;
    scanf("%d",b); //this scanf means go at b's address and
    printf("%d", a+50);
}
```

Value at that address change which we will enter through scanf//

Sol^n:-



Suppose we Enter → (70)

↑ It means go at 6th address & at 1000

& Change the value into (70) ✓

so O/P ⇒ 70 + 50 → (120) Ans ✓

Que

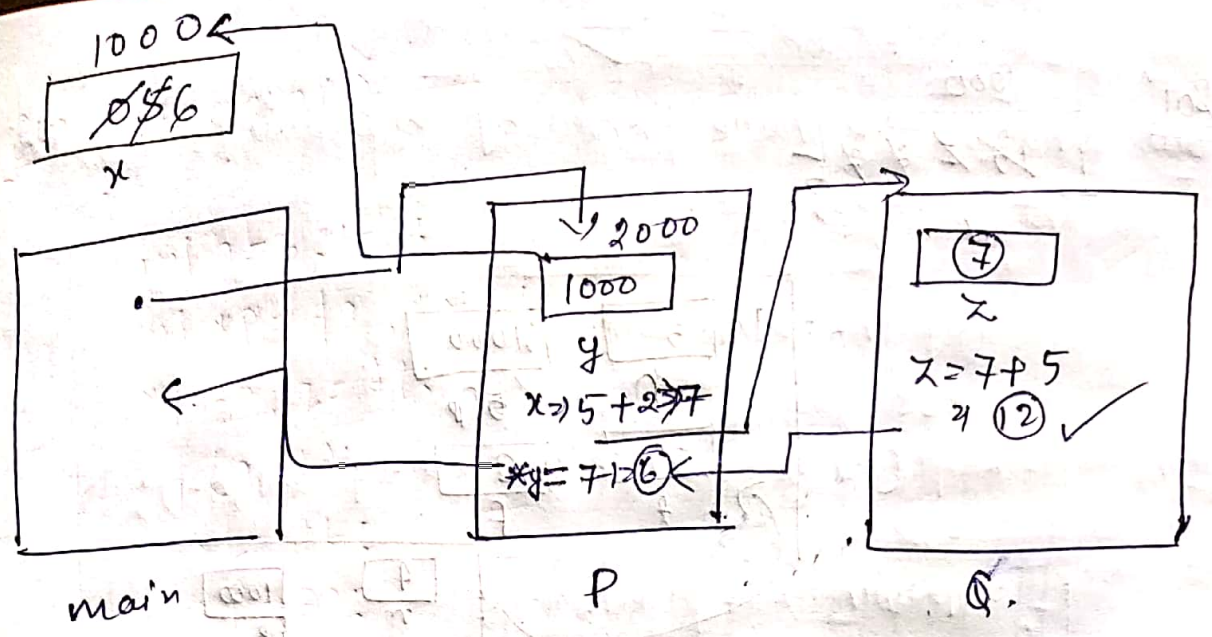
```
int x;
main()
{
    x=5;
    P(&x);
    printf(x);
}

Void Q(int z)
{
    z+=x;
    printf("%d", z);
}

Void P(int *y)
{
    int x = *y+2;
    Q(x);
    *y = x-1;
    printf("%d", x);
}
```

O/P of this program is=?

(a) 127, 6 (b) 22, 12, 11

(c) 14, 6, 6 (D) 7, 6, 6

Scanned with CamScanner



12, 7, 6 Ans (a)

Que
```
int main()
{
    int x=15;
    printf("%d", fun(5,&x));
    return(0);
}
```

```
int fun(int n, int *fp)
{
    int t, f;
    if (n<=1) { *fp=1; return(1); }
    t= fun(n-1, fp);
    f= t+ *fp;
    *fp =t;
    return(f);
}
```

The value printed is

(a) 6 (b) 8

(c) 14 (d) 15

P.T.O

Note → For any operand if unary & Binary operator come immediate to the given operand then occurance priority of unary operator is more than occurance of Binary operator

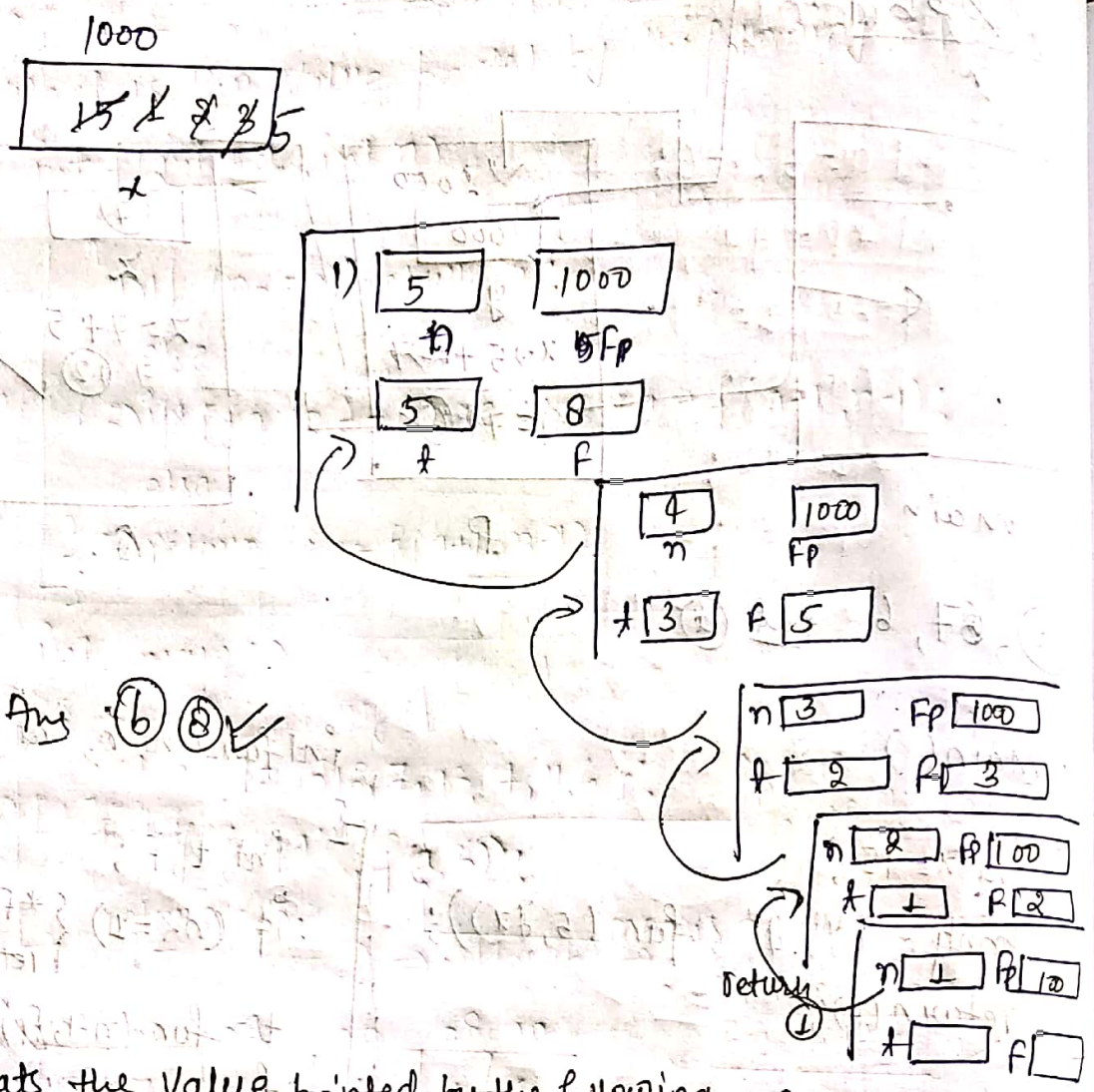
Ex → y=(*x)+y

→ but unary (*) will win

→ unary & Binary operator Both Fight for x is mine

Scanned with CamScanner

Sol^n



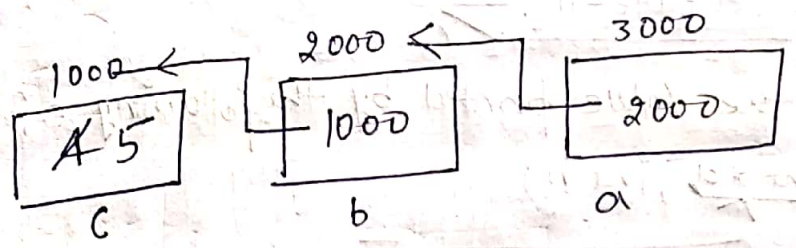
Ans (b) (8)

Q^{ed} Whats the value printed by the following C-program?

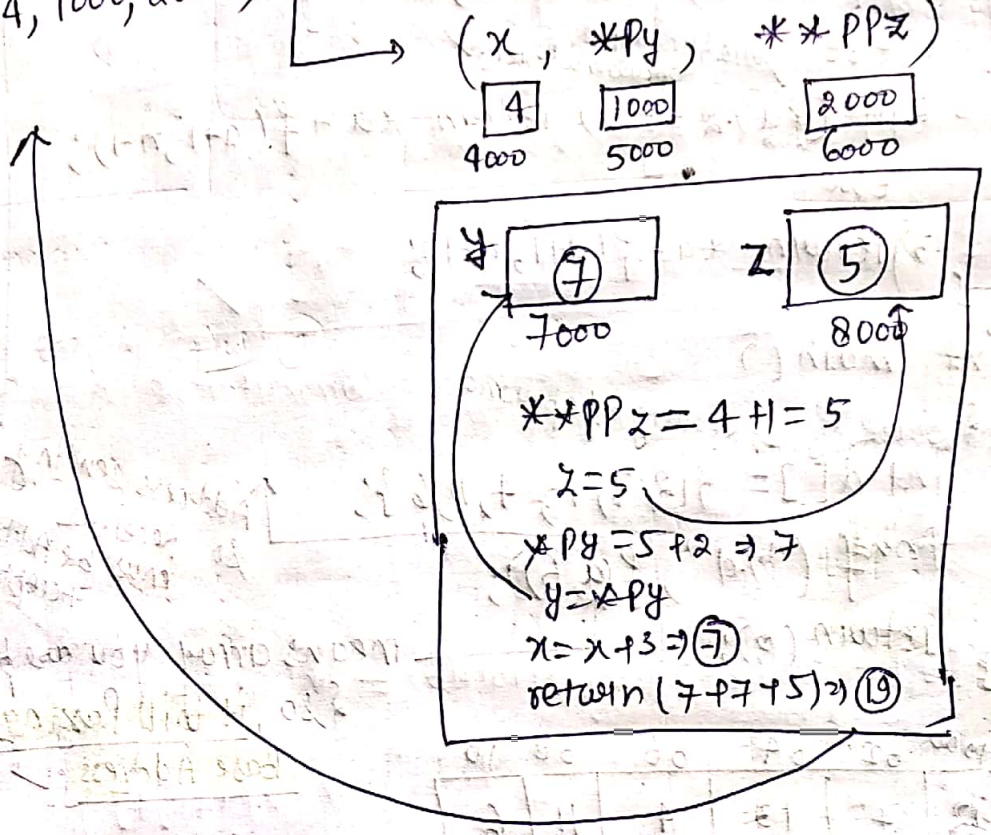
```
Void main()
{
    int c, *b, **a;
    c = 4;
    b = &c;
    a = &b;
    Printf("%d", f(c, b, a));
}
```

```
int f(int x, int *py, int **ppz)
{
    int y, z;
    **ppz += 1;
    z = **ppz;
    *py += 2;
    x += 3;
    y = *py;
    return (x+y+z);
}
```

Soln ✓



f (4, 1000, 2000) → (x, *py, **ppz)



Ans 19 ✓

Scanned with CamScanner

Lecture-11 :

Que What is the value printed by the following C-Program?

```
int f(int *a, int n)
{
  1) if (n<=0) return(0);
     else
  2) if (*a % 2 == 0) return *a + f(a+1, n-1);
     else
  3) return *a - f(a+1, n-1);
}
int main()
{
  int a[] = {12, 7, 13, 4, 11, 6};
  printf("%d", f(a, 6));
  return (0);
}
```

win unary operator winner

① → Reading of operand

② → Address sending

So Receiver must be Address Acceptor (Pointer Variable)

means array you are passing so, it will Passing only Base Address

Sol^n

a	1000	02	04	06	08	10
	12	7	13	4	11	6
	0	1	2	3	4	5

a	n
1000	6 5
2000	3000

f(int *a, int n)

1)

2) 12%2=0 → return(12 + f(1002, 5)) ③

a	n
1002	5
4000	5000

15

3) 7 - f(1004, 4) → ③

a	n
1004	4
6000	7000

3) 13 - f(1006, 3) → ④

2) 4 + f(1008, 2) → ⑤

3) 11 - f(1010, 1)

2) 6 + f(1012, 0) return 0

Ans ⇒ 15

Scanned with CamScanner

Concept

a[]={1,2,3} OK

a[3]={1,2,3} OK.

But only a[] ← Not OK

- Purpose of declaration is creating memory.
- Priority [] > int

- Array Name means → Base address.

If

① a

1	2	3	4
1000	02	04	06

→ Printf(a) ⇒ 1000

a[3] ≡ *(a+3)

② a [2] (1000) → Printf(a) ⇒ 2

In array (a+1) → means in array skip 1 element & a means Base Address

Ex

	1000	02	04	06	08
a	12	7	13	4	11
	0	1	2	3	4

→ To access this value

*(a+3)

↳ in array skip 3 people

↳ value at array (a) after skipping 3 people

→ can be written as.

→ a[3]

Pf(a) = 1000

Pf(a+1) = 1002

Pf(a+2) = 1004

Pf(a+4) = 1008

→ if we write

a = a+2

↳ means (1000 + 2*2) ⇒ 1004

1000 ← Not same → 1004

So, In array (a = a+2) → Not Possible

↳ Error message given by system.

Note :-

↳ Base address of array cannot be change.

↳ we can say L value cannot be change

↳ a = a+2

L-Value → a ← R-Value

- $\boxed{a[i] \neq a+i}$ { Bcoz $(a+i)$ ← gives address after skipping i element in array a }

{ $a[i]$ ← gives value at ith place of array i }.

★★★

So ⇒ $\boxed{a[i] = *(a+i)}$

Note ① ★★★

$\boxed{a[i] = *(a+i) = *(i+a) = i[a]}$ ✓

↳ addition are commutative in Nature ✓

Note ②

b ← assume

$a[i][J]$

⇓

$b[J] \Rightarrow *(b+J) \Rightarrow$ Replace(b) $\Rightarrow *(a[i]+J)$

⇓

$*(*(a+i)+J)$

★★★

So $\boxed{a[i][J] = *(*(a+i)+J)}$ ✓

- Base Address of Array * Cannot be change. { L-Value cannot change }
- In arrays $a+i$ = means skipping i element.
- Passing Array means passing Base Address ✓

Scanned with CamScanner

Que main()

```
{
  Char a[]= "programming 2020"; // string storing
  Printf ("%s", a+a[3]-a[5]);
}
```

Sol^n

a	p	r	o	g	r	a	m	m	i	n	g	2	0	2	0	\0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Tells string over

String always End with null Character

- In the Question string length = 15
  But If we are storing then string size will be 16

on Extra space for Null \0 Character

String size = string length + 1 → one Extra for Null character.

How to read ~~a[]~~ Char a[16] → a is a array having 16 slots & each slot contain character type value.

Concept

Strlen("abcd") ⇒ 4

Size of ("abcd") ⇒ 4 + 1 ⇒ 5 (+ for Null)

Concept ↳ If you want to store a string without using Null character then store it by using single quotes {store one-one character}

Ex Char a[] = {'a', 'b', 'c', 'd'}

→ Printf ("%s", a+a[3]-a[5]);

1000 + g − a

1000 + 103 − 97.

1000 + 6 ⇒ 1006 ← it means Print string from add. 1006 onwards upto Null Character.

(mming 2020)

Concept of string

- Printf("%s", "abcd") ⇒ abcd → string directly give
- ** Printf("%s", 1000) ⇒ abcd // then it will Print from Base address & upto Null character // ← Base Address give
- Printf("%s", 1012) ⇒ // from Address 1012 to Null character it will Print //
- printf("%s", a) ⇒ // It means we will give Base Address of Array a

*** $\boxed{\text{Printf("\%s", 1000)} \equiv \text{Printf("\%s", a)}}$

If it is Base Address of Array a

directly Base Address give

- Printf("%s", a+5)

1000 + 5 → 1005 → so, print from address 1005 till null.

each slot of ~~Array~~ character array will be 1 Byte

- Printf("%c", *(a+5))

at (1005) ← only that character value print

- Printf("%d", *(a+5))

→ ASCII value at (1005) will print

ASCII Value

A → 65

a → 97

0 → 48

Que main( )

```
{
    char a[10];
    char *b = "string";
    int length = strlen(b);
    for (i=0; i<length; i++)
    {
        a[i] = b[length-i];
    }
    Printf ("%s", a);
}
```

*** This string store some where Inside system & it's Base Address given to Variable "b"

|| → "b" is a Variable which contain Address of character So b is a Character Pointer.

So Here First Character Address of string will be given to Variable b.

b → 3000 (2000)

• Last me Null to hega Chahe wo Array me store ho ya hai generally string

3000 01 02 03 04 05 06

Sol^n

a	1000	01	02	03	04	05	06	07	08	09
	\0	g	n	i	r	t				
	0	1	2	3	4	5	6	7	8	9

length = 6 ✓ Fill

b = "string" 0 1 2 3 4 5 Fill

(6<6) → Loop failed ✓ So, "s" will not be stored ✓

i = 0 ↓ a[0] = b[6-0] ⇒ b[6] | i = 1 ↓ a(1) = b(6-1) ⇒ b(5) | i = 2 ⇓ a(2) = b(6-2) ⇒ b 4 ✓

- - - - →

Printf ("%s", a) ⇒ Nothing {Blank} ✓ → No output

↳ Starting Null character ✓ so stop ✓

** By default array Contain Garbage Value. → also pointers & Local Variables ✓✓

*** ✓✓ • In string if we Not put Null character at end then our program will go to ∞ loop. ✓✓

Q4 main ( )

```
{
  int i;
  for (i=1; i<30; i++)
  {
    switch (i)
    {
      case 1: i+=7;
              i+=8;   → No break so continue your Execution
      case 3: i+=2;
      case 10: i+=5;
      default: i+=3;
               break;  → Break Here ✓
    }
    printf(i);
  }
```

Soln

i = 1 → 8 → 11 → 13 → 18 → 21 → 22 → 25 → 26 → 29 → 30 → 33 → 34

1000

O/P

21
25
29
33 ✓

→ If any case is matching then default case will not be printed. {when break after each case is given otherwise matching case se leky Break TK { Break Include} sabhi Print honge.

Def. Default case will be printed only when No case is matching.

***Que**

```
main ()
{ int i=-1, j=-1, k=-1, l=2, m;
  m=((i++ && j++ && k++) || (l++));
  printf (i,j,k,l,m);
}
```

Post Increment operator ✓

So, Variable will be Incremented after work done of that line

Sol^n :-

m = ((i++ && j++ && k++) || (l++))

i = -1 → True, j = -1 → True, k = -1 → True

True

→ True → 1

* Any Non zero Value will be known as True ✓

→ ~~After the~~ After this line

$i = j = k = 0$ → due to Post Increment

O/P 0, 0, 0, 2, 1

[will will Not execute (l)] → This property is known as short circuit of programming language

BCOZ

True || Anything → True (Always) ✓

No Need to Check it { K/a short circuit of programming language }

Shortcircuit Concept

~~0 || Any~~ 1 || Anything → Anything Not Need to check for Result (→ or operator ✓)

0 || Anything → Anything Need to check

~~1 &&~~ 1 && Anything → Anything Need to check

0 && Anything → Anything Not Need to check

Scanned with CamScanner

Lecture – 12

★★★★★ **Question**

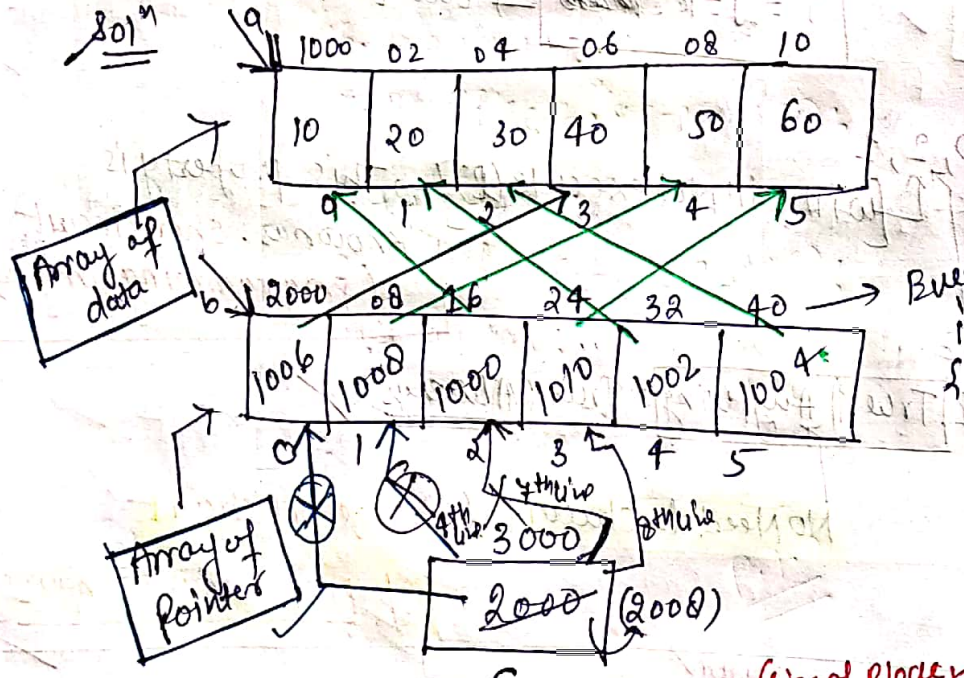
```
main()
{
1) int a[] = {10,20,30,40,50,60};
2) int *b[] = {a+3, a+4, a, a+5, a+1, a+2}; // Array of Pointers
3) int ** c = b; // int ** c = b
4) **++c; // c Point to Next Block
5) printf(c-b, *c-a, **c);
6) **c++;
7) printf(c-b, *c-a, **c);
8) *++*++c;
9) Printf(c-b, *c-a, **c);
}
```

① int ** c = b

c is a Variable which store double Pointer

c is a double Pointer

Solⁿ



Every one is 8 Byte {Assume Addresses are 8 Bytes}

(Size of Block Pointer)

c-b ⇒ 2008 − 2000 ⇒ 8 → 8/8 ⇒ ①

*c-a ⇒ 1008 − 1000 ⇒ 8/2 ⇒ ④

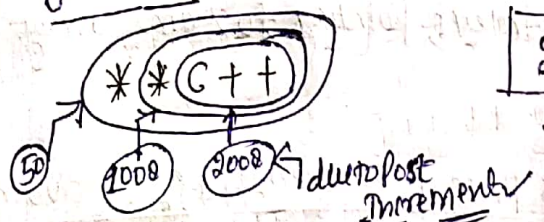
**c ⇒ 50

If you are subtracting larger pointer to the smaller pointer then it will gives # elements between them excluding Last Pointer But including First pointer.

5th line prints → 1, 4, 50

from Previous Page

6th Line



*** Priority

Unary Operator	Binary Operator

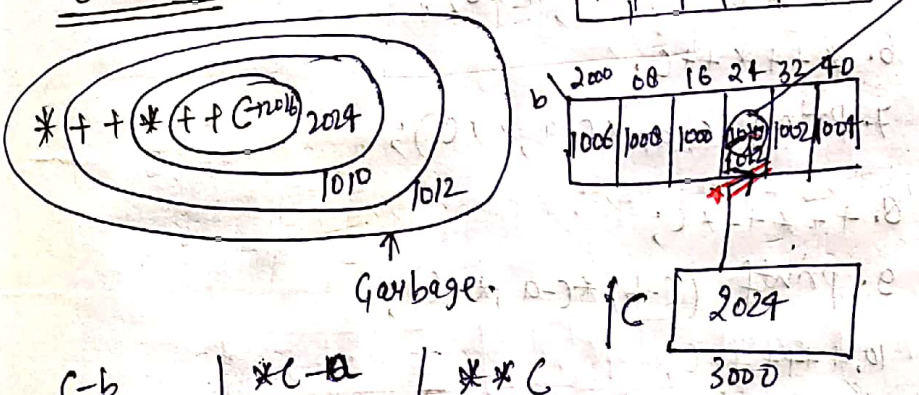
→ Heard priority Table

** A++ > * ← Priority

7th Line // at 7th line c address change to 2016 from 2008

c-b 2016 − 2000 ⇓ 16/8 ⇒ 2	*c − a 1000 − 1000 0/2 ⇒ 0	**c (10)

8th Line :-



c-b 2024 − 2000 ⇒ 24/8 ⇒ 3	*c − a 1012 − 1000 12/2 ⇑ 6	**c Garbage

Note :-

① Priority for * & ++ ⇒ ++ have Heighest priority than *

② Assume p is a pointer ⇒ (P+5) means → {skip 5 element from P}

(or)

(P+1) means → skip 1 element from P in forward direction.

③ $(P - i) \rightarrow$ skipping i element from P in Backward direction ✓

④ $(P_2 - P_1)$ Assume P_1 & P_2 are pointers, pointing to same array.

and P_1 & P_2 ⇒ $\boxed{P_1 < P_2}$ ✓

then $P_2 - P_1$ = # elements present between P_1 & P_2, including P_1 but Excluding P_2. ✓

Que :- main()

```
{
1. float a[ ]={10,20,30,40,50,60};
2. float *b[ ]={a+2,a+3,a,a+1,a+5,a+4};
3. float **c=b;
4. *++*c++;
5. printf(c-b,*c-a,**c);
6. *++*++c;
7. printf(c-b,*c-a,**c);
8. ++*++*c;
9. printf(c-b,*c-a,**c);
10. ++++**c;
11. printf(c-b, *c-a,**c);
}
```

Sol^n

P.T.O

Sol^n

a

1000	04	08	12	16	20
10	20	~~30~~ 33	40	50	60
0	1	2	3	4	5

Line ① →

b

2000	08	16	24	32	40
~~1008~~ 1012	1012	~~1000~~ 1008	1004	1020	1016
0	1	2	3	4	5

Line ② →

Line ③ → 3000 | ~~2000~~ ~~2008~~ 2016 | ← in line 5 ← It will change ✓

c

← line ⑥

Line-④

* ++ * (c 2000 ++) 2000 1008 1012 (40) ✓

Line-⑤

c − b	*c − a	**c
~~2008 − 2008~~	1012 − 1000	(40) ✓
→ 2008 − 2000	⇒ 12/4 ⇒ (3) ✓	
→ (8)/8 ⇒ (1) ✓		

Line ⑥

* ++ * ++ (c 2008) 2016 1000 1004 20

Line ⑦

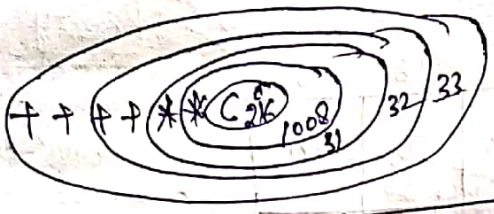
c − b	*c − a	**c
2016 − 2000	1004 − 1000	(20) ✓
↳ 16/8 ⇒ (2) ✓	↳ 4/4 ⇒ (1) ✓	

Line ⑧

++ ** ++ * (c 2016) 1004 1008 30 (31) ✓

c − b	*c − a	**c
↳ 2016 − 2000	↳ ~~2016~~ 1008 − 1000	↳ (31)
⇒ 16/8 ⇒ (2)	⇒ 8/4 ⇒ (2)	

Scanned with CamScanner



c-b	*c-a	**c
↓	↓	↓
2	2	33

***

main ()

array of storing Address which is character

But you find string it means that string storing at some where into system & you keep first character address to str

{

1) Char *a[] = { "bacha", "Paapaa", "go to hell", "brainless", "buffalow" };

2) Char **b[] = { a+2, a+3, a, a+4, a+1 };

3) Char ***c = b;

4) *++**++c;

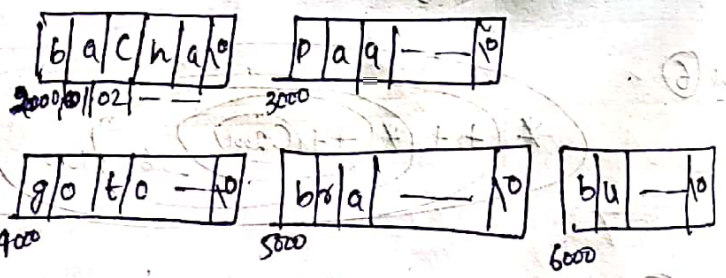
5) Printf ("%s", ++***c+1);

6) printf ("%c", *++***c+2);

7) printf ("%s", **(c+2)+2);

}

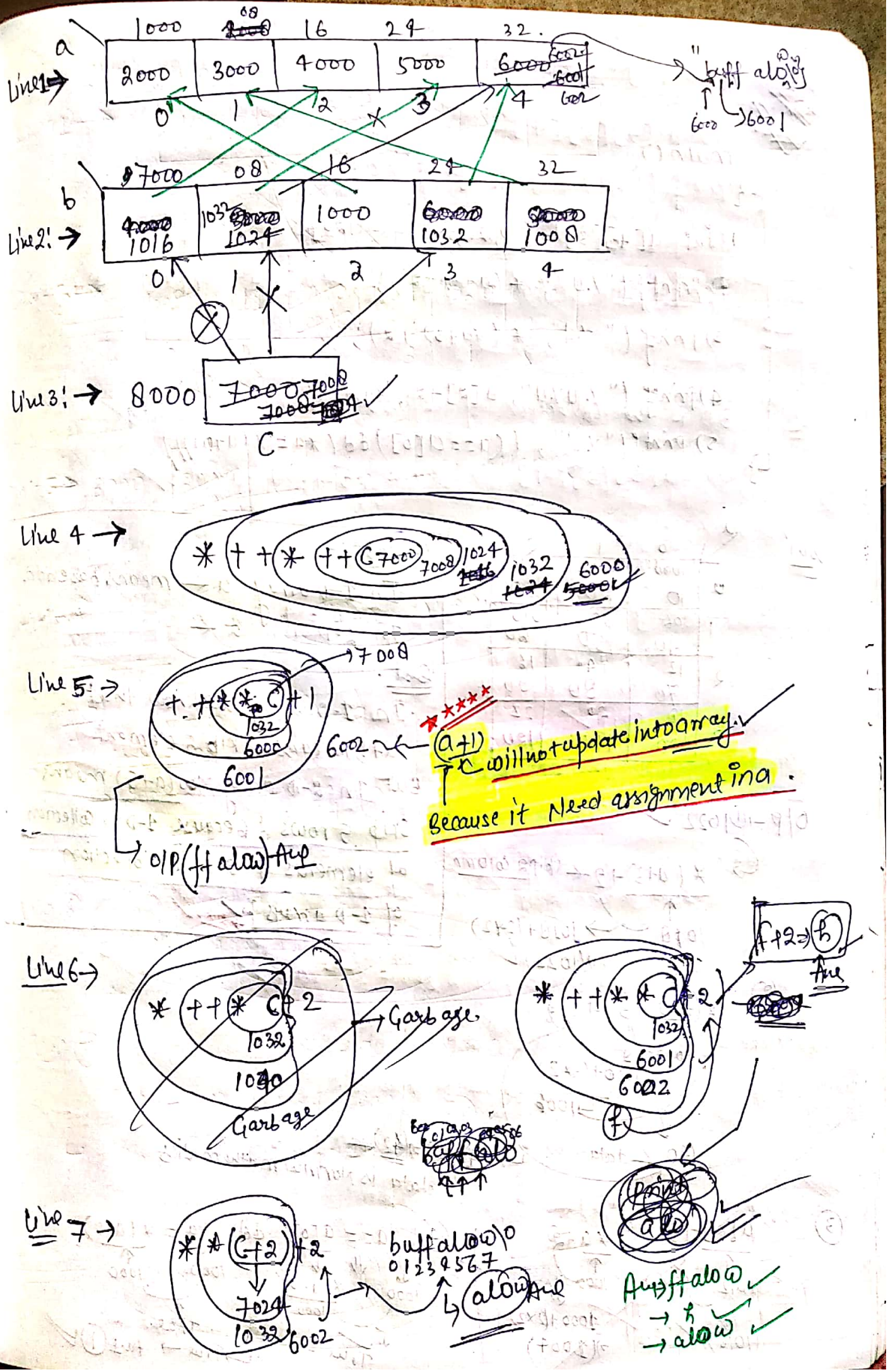
Sabhose storing of Array



So,

a	1000	2008	16	24	32
	2000	3000	4000	5000	6000
	0	1	2	3	4

P.T.O



Scanned with CamScanner

-: Lecture-13 :-

***Que:-**

```
main()
{
1) int a[4][3] = {10, 20, - - - - 120}
2) printf("%u", *(a+3)+2);
3) printf("%d", *(*a+3)+2);
4) printf("%u%u", a[2]+2, *a+2);
5) printf("%d", ((a==a[0]) && (*a=*(a+0))))
}
```

Sol^n

a	0	1	2
0	1000 10	1002 20	04 30
1	06 40	08 50	010 60
2	12 70	14 80	16 90
3	18 100	20 110	22 120

Concept:-

In 1-D array a ← means base add

In 2-D " a ← " " " "

~~But~~

- In 1-D array we know that (a+5) means skip 5 element

But In 2-D array (a+5) means skip 5 rows {Because 1-D is collection of elements & 2-D is collection of 1-D arrays} ***

O/P → ① 1022

*(a+3)+2 ← skip 2 colomn

1018 → 1018+(2*2) ⇒ (1022)

② *(*a+3)+2

1000 → 1000+(3*2) → 1006

(40) ← data → (40+2) ⇒ (42)

↳ data is Normal mathematics

③ a[2]+2 | *a+2

1012 + (2×2) | 1000 → 1000+(2×2) ⇒ (1004)

(1016)

④ a==a[0] && *a = *(a+0)

1000 ⇓ *(a+0) 1000 | 1000 1000

True && True → True → Ans ①

Scanned with CamScanner

*** Concept of 2D-Arrays :-

a	0	1	2	3
0	1000 16	02 17	04 18	06 19
1	08 20	10 21	12 22	14 23
2	16 24	18 25	20 26	22 27
3	24 28	26 29	28 30	30 31

⇒ "a" ← means Base add.

(a+1) ← means one row skip ✓

*(a+1) ← one row skip but Here star means second row selected.

(*(a+1)+3) ← means one row skip & select second Row, In that second row three column skip further

((a+1)+3) ← means one row skip & select second row, Three column skip & then select 4th column ✓

Also equal to a[1][3] ⇒ 23 ***

In this way

(a) ← means you are in 0th row but you are not selected

*(a) ← 0th row selected

**(a) ← 0th row & 0th column selected ✓

**(a) means a[0][0]

↳ ≡ *((a+0)+0) ✓

Continue

(a+1) ← skip 1 row ✓

*** ((a+1)+2) ← means after skipping 1st row. skip 2 more Row.

*** *((a+1)+2) ← means select 3rd row.

*** (*((a+1)+2)+2) ← In 4th row skip 2 column ✓

(((a+1)+2)+2) → select 4th row & select 3rd column {after skipping 2 columns}

After skipping 3 row

⇓

It is just look like

a[3][2]

Scanned with CamScanner

***Que** main[]

```
{ int a[5][3][4]
  printf("%u %u %u", a[2]+2, *a[3]+3);
  printf("%u %u %d", (a+2)+2, *a+2+2);
  printf("%u %u %u %d", *a[2]+2, *a[3]+7, *(**(a+4)+2)+7)
}
```

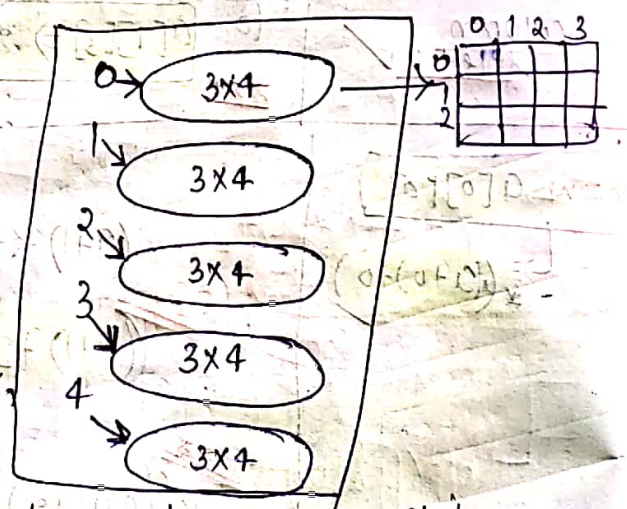
Sol^n 3D is the collection of 2D's ✓

Here a[5][3][4] → means 5→(3×4) array ✓✓

↓

Here

- a ← Base Add ✓
- a+1 ← one (3×4), 2D skip means 12 element skip ✓



*(a+1) ← means one 2D skip & select second 2D ✓

{*(a+1)+2} ← means one 2D skip & select second 2D. In that second 2D 2 Rows skip.

((a+1)+2) ← Select 2nd 2D. Then skip 2 Row & select 3rd row ✓ of 2nd 2D ✓

(*(*(a+1)+2)+3) ← select 2nd 2D
↓
then select 3rd row of 2nd 2D
↓
then select 4th column of 3rd row of 2nd 2D ✓

***

((*(a+1)+2)+3) ← select 2nd 2D
↓
Select 3rd row
↓
Select Fourth column & then select it's Value

⇓

same as

$\boxed{a[1][2][3]}$ ✓

Scanned with CamScanner

• In 3-D you does not find the value untill 3 star exist
& In 2-D you does not find the value untill 2 star exist

V.V Imp

In 3D
a[2]+3 → (*(a+2)+3)

⇓

one [ ] means one(*)

Concept:-

*** ((a+1)+1) ← ~~select 2D~~ skip 2 time 2D

*** *((a+1)+1) ← select 3rd 2D

a[1][4] ← shortcut & Actual

↓

((a+1)+4)

Sol^n

• a[2]+2 → *(a+2)+2

↓

⇒ 1000+{(3×4)×2+(2×4)}2

⇒ 1000+(32)×2

⇒ 1064

• **a[3]+3 → **(a+3)+3

⇓

⇒ ((12)*3+3)*2+1000

⇒ (39)×2+1000

⇒ 1000+78

⇒ 1078 Ans

After skipping 3, 2D array select 3rd 2 array & then select 1st row of that array then skip 3 column

• (a+2)+2 ≡ (a+4) → 4, 2D skip

*{a+4}

⇒ 1000+{4*{12}}*2

⇒ 1000+(48*2)

⇒ 1000+96 ⇒ 1096

• *a+2+2 ≡ (*a+4)

Select first 2D array then skip 4 Rows {each row have four element}

⇒ 1000+(4*{4})*2

⇒ 1000+(16)×2

⇒ 1032

• *a[2]+2

⇒ select after skipping 2 ~~arrays~~ 2D array

also select first row of 3rd 2D array

⇒ {(12*2)+(2×4)}*2+1000

⇒ (24+2)*2+1000

⇒ 52+1000 ⇒ 1052

• *a[3]+7

⇓

{(a+3)}+7

⇒ {1000+(36×2)}+(7*2)

⇒ 1000+72+14

⇒ 1000+86 ⇒ 1086

((*(a+4)+2)+7)

↓

1000+(4*12)×2

1096

1096

↓

1096+4

1100

↓ Garbage+7

⇒ Garbage

In array

1D → Collection of element
2D → Collection of 1D
3D → Collection of 2D

V. Imp

In 1D → (a+1) means → skip one element
" 2D → (a+1) means → skip one 1D array
" 3D → (a+1) means → skip one 2D array

***

Que

1) int a1[ ] = {6, 7, 8, 18, 34, 67};
2) int a2[ ] = {23, 56, 28, 29};
3) int a3[ ] = {-12, 27, -31};
4) int *a[ ] = {a1, a2, a3};

5) main()
{
f(x);
}

6) f(int **a)
{
printf("%d", a[0][2]);
printf("%d", *a[2]);
printf("%d", *++a[0]);
printf("%d", *(++a)[0]);
printf("%d", a[-1][+1]);
}

***

**a → double ** because one * shows its store address & second * shows inside that address one more address present go to that & find values

What is the out put of the above program?

PP.O

*(++a)[0]

((4008 + 0))

2000

23 Ans — 4

a[-1][+1]

((a+-1) + 1)

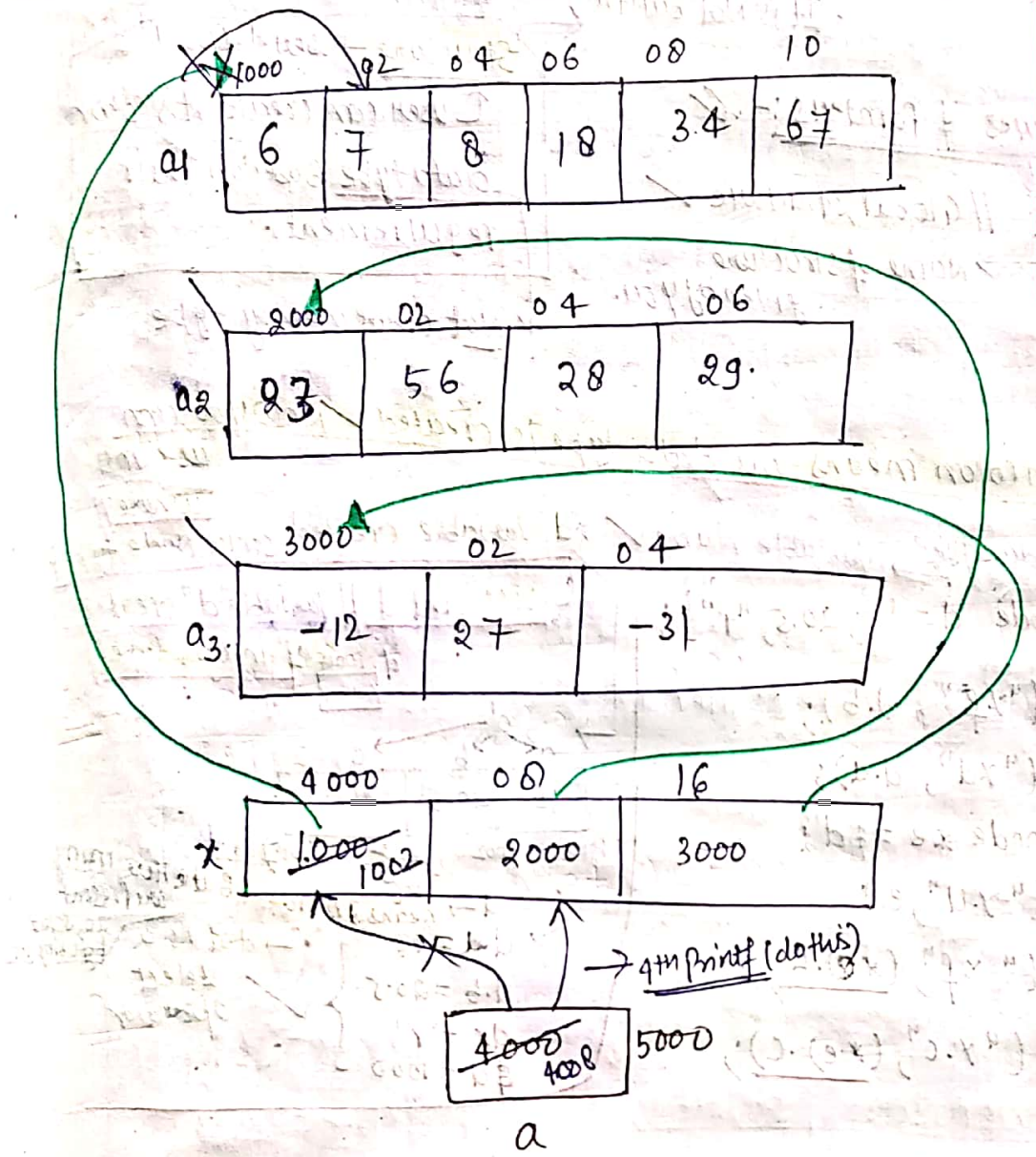
4008

4000

1002

1004

8 Ans — 5



★★★★★
Concept ✓

a+2
↑ a will not update

a=a+2
(or)
a++
"a" will update

a[0][2] ⇒ Always convert shortcut into Actual form to find the values ✓

→ a[0][2] ⇒ *(*(a+0)+2) → 8 Ans ✓ — 1

*a[2] ⇒ *(*(a+2))

*++a[0] ⇒ *++*(a+0)

((a+2))

++(a+0)
4000
1000
1002
→ 7 Ans ✓ — 3

((a+2))
4000
4016
3000
→ -12 Ans ✓ — 2

Scanned with CamScanner

· It is Not builtin data type

Structure → user defined
(user can create it's own data type Based on their requirement.

↓ Not create memory type

Structures & Pointers :-

always start from here

```
struct node   // Global structure
{
    int a;
    float b;
    char c;
};   // semicolon means one data type is created ✓ Now you can use any where //
```

→ keyword

→ Name of structure given by you

main()

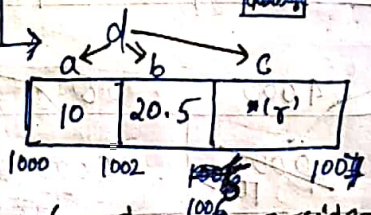
```
{
① struct node d = {10, 20.5, 'r'};
② Printf ("%f", d.b);
③ Printf ("%d", d.a);
④ struct node *e = &d;
⑤ printf("%u", e);
⑥ printf("%f", (*e).b);
⑦ printf("%c", (*e).c);
}
```

→ datatype, → Variable Name ✓

{"d" Variable created struct Node type}

↳ Like int d // Variable "d" created of integer type //

d 1000



d → means Inside → But Here many are Present

d.a = 10
d.b = 20.5
d.c = 'r'
&d = 1000

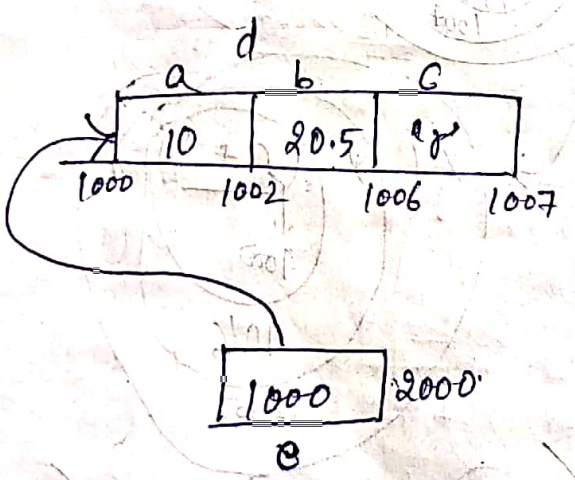
"." → dot is a select Operator ✓

O/P ⇒ 20.5, 10, 1000, 20.5, 'r' ✓

Line-④ struct node *e = &d;

Pointer Variable e created which contain Address of struct Node type :-

Figure



e → 1000

*e → Inside "d" you go

There are many people So select one of them.

{(*e).b} → 20.5

⊙ If you know Integer Variable then you know structure about Variable ✓ only one thing is more in structure Variable is create structure & then use it ✓

Inside structure you can use user defined data Type ✓

Que

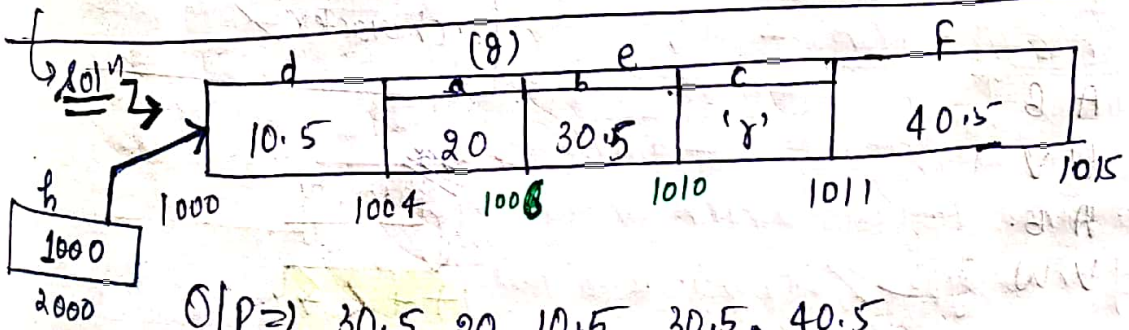
```
struct S1
{ int a;      → 2B
  float b;    → 4B   (7B)
  char c;     → 1B
};

struct S2
{
  float d;    → 4B
  struct S1 e; → 7B   (15B)
  float f;    → 4B
};
```

Variable e created of struct S1 type ✓✓

```
main()
{
① struct S2 g = {10.5, {20, 30.5, 'r'}, 40.5};
② printf("%f", g.e.b);
③ printf("%d", g.e.a);
④ printf("%f", g.d);
⑤ struct S2 *h = &g;
⑥ printf("%f", (*h).e.b);
⑦ printf("%f", (*h).f);
}
```

goto Inside h means go Inside to Add 1000 ✓✓

★★★ If Inside structure Again structure then To access Value double dot Required ✓✓



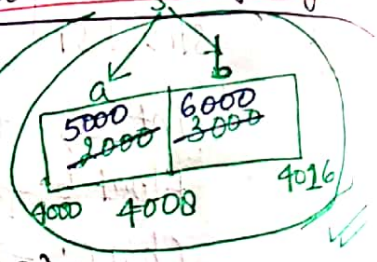
O/P ⇒ 30.5, 20, 10.5, 30.5, 40.5

Scanned with CamScanner

- For Pointer → * required
- For structure → . required

- (*e).c means → (e→c)
- ((*h).e.b) means → ((h→e).b)

{Structure & Integer} Both are same
Integer means only one variable going
Structure means more then one variable going



Que (good Question) Consider the following C-Program.

① struct node.
{
char *a; ⇒ 8B
char *b; ⇒ 8B } 16B
};
→ by default "t" is a struct Node data type.

② typedef struct node t; //renaming

③ main()
{
t s = {"A", "B"};
printf("%s%s", s.a, s.b);
f1(s);
printf("%s%s", s.a, s.b);
f2(&s);
}

one variable names of struct Node Type is created
"A" → 5000, "B" → 3000, → string
t means Structnode Address.
"s" go means Inside's" is going (2000, 3000)
1000 going so, Receiver will be Pointer
&s → 1000

④ f1(t s)
{
s.a = "U"; → 5000 (address)
s.b = "V"; → 6000 (address)
printf("%s%s", s.a, s.b);
}

⑤ void f2(t *p)
{
(*p).a = "V"; → 8000
(*p).b = "W"; → 9000
printf("%s%s", (*p).a, (*p).b);
}

p: 1000, 7000 — Structnode Type Pointer

a	b
~~2000~~ 8000	~~3000~~ 9000

Sol^n

a	b
~~2000~~ 8000	~~3000~~ 9000

1600, 1008, 1016

giving string means given add. of first character

O/P ⇒ A B
U V
A B.
V W

Priority
++ <->

P→c means (*P).c

Scanned with CamScanner

*** Que

good Question

① struct Test

```
{ int i; → 2B }  10B
  char *c; → 8B
};
```

→ st is a array of 5 slots. which is struct Test Type ✓

② struct Test st[5] = { {5, "become"}, {4, "better"}, {6, "Jungle"}, {8, "ancestor"}, {7, "brother"} };

(1 → 2000, 2 → 3000, 3 → 4000, 4 → 5000, 5 → 6000)

③ main ()

```
{ struct Test *p = st;
  p = 1;
  ++(p→c);
  printf("%s", (p+1)→c);
  printf("%c", *(++(p→c)));
  printf("%d", p[0].i);
  printf("%s", (p→c));
}
```

(p→c: 3000 → 3001; (p+1): 1010, 3001; ++(p→c): 1002, 4001 → value at 4001; p[0].i → (*(p+0)).i → *(1000+0).i → 6; p→c → 4001 ✓)

only one Person Fighting ✓

Sol^n

	0	10	20	30	40
	i, c	i, c	i, c	i, c	i, c
	5, 2000	4, 3000 → 3001	6, 4000 → 4001	8, 5000	7, 6000
	0	1	2	3	4
	st[0]	st[1]	st[2]	st[3]	st[4]

7000 [1000 ~~3000~~ ~~4000~~] 1000

p ← pointer so it will be 8 Byte & containing struct Test Type address

O/P:- etter
U
6
ungle ✓

Lecture - 14 :-

→ Converting from one Type To another type ✓

Type casting in Pointers :-

Good Que

```
main ()
{
1. int a[10] = {300, 301, 302, ---- 310};
2. Char *b = (Char*) a;
3. int *c = a;
4. float*d = (float*) a;
5. void *e = a;
6. Printf( b, c, d, e);
7. Printf( *b, *c, *d, *e);
8. Printf (b+1, c+1, d+1, e+1);
}
```

Sol^n

301 store into Binary formate

302

310

a	1000	1001	1002	1003	1004	1005		1008	1009
	00101100	00000001	00101101	00000001	00101110	00000001	----	00110101	00000001
	0		1		2		----	9	

300 → (0000000 | 00101100)

lower Byte store in lower Address {Called Little Enclian formate} ✓

↳ By default we use little enclian formate.

Line ② → Char *b = (Char *) a;

↳ after type casting it will be character Address

it is saying that I ~~con~~am a pointer variable which contain Address of char type ✓ {Character Pointer Variable} ✓

"a" is a Integer Address but b want to store character Address. So do type casting of a so that we can store a easily.

↳ ye a aur b ko same type me change kr dega.

Scanned with CamScanner

concept

char *b = (char *)b
↑ Type cast to character pointer

char b = (char)b
↑ Type cast to character data

Char *b = (Char *) a → Before type casting it will be Integer Address
1000 1000

After type casting → Char Add → size of Both are same But ~~Name~~ type changes ✓

b → char pointer
Character Address → 1000
2000

c → integer pointer
1000 → Integer Add.
3000

Line → 3 → int *c = a → Both side same ~~side~~ Type so No problem so Here No Type casting occurs.
"c want Address of integer type ✓

Line → 4
Float *d = (float*) a → ∵ a is a Integer Add, so it have Integer Address
d want Address which is Float Type
So, Type casting done
↓
d → Float type pointer
1000
4000

Line → 5 Void *e = a → No type casting Required e accept any type Address I can accept ✓
→ A gives Integer type Address ✓
Any type
It is saying that I want address of Any type

To type pointer → e
1000
4000

from above all ~~b, c~~, b, c, d & e are pointing to 1000 of array a only

Scanned with CamScanner

O/P

Print
- b → 1000
- c → 1000
- d → 1000
- e → 1000

} line 6

Line 7

*b → ~~It is~~ "b" is a character Pointer & if we want to print value of character pointer then it will print 1-1 byte data only

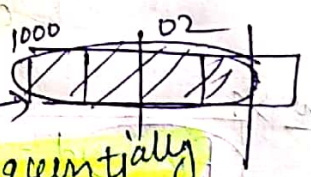
↳ will print → 00101100 → 44

*c → It is integer Pointer

↳ 2 Byte data print → 300

*d → It is a Float Pointer

↳ 4 Byte data it will read sequentially



*e → Void Pointer

↳ error message will come

{It dont have ~~not~~ any type means It have not any clearity}

Line 8

↳ (b+1) → It is a character Pointer so 1 byte will go further

character Pointer

1000 → 1001

(c+1) → 1002

1000

(d+1) → 1004

1000

(e+1) → error

1000

• all pointer storing time take same size

But when we Increment Address,

Char → 1B move

Int → 2B "

Float → 4B "

Void → error

Scanned with CamScanner

*** Void pointer → Storing Address time No problem create but Retrieving Time ~~to~~ problems create

Concept

previously → e is a Void Pointer

if we type cast it like → (int*)e

int *p

then we can access the data (Retriving time do type casting).

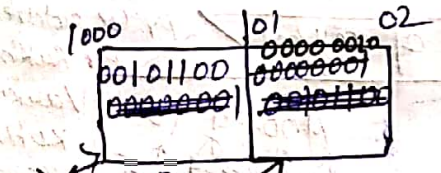
Que main()

```
{ int i = 300;
  char *ptr = (char*) &i;
  *++ptr = 2;
  Pf("%d", i);
}
```

what's the output?

(a) 556 (b) 300 (c) 655 (d) 003.

Soln



300 → 00000001 00101100

ptr

char *ptr = (char*) &i

1000

1000

*++ptr = 2

→ 0000 0010 0010 1100

512 32 8 4

556 ✓

556

Good Que

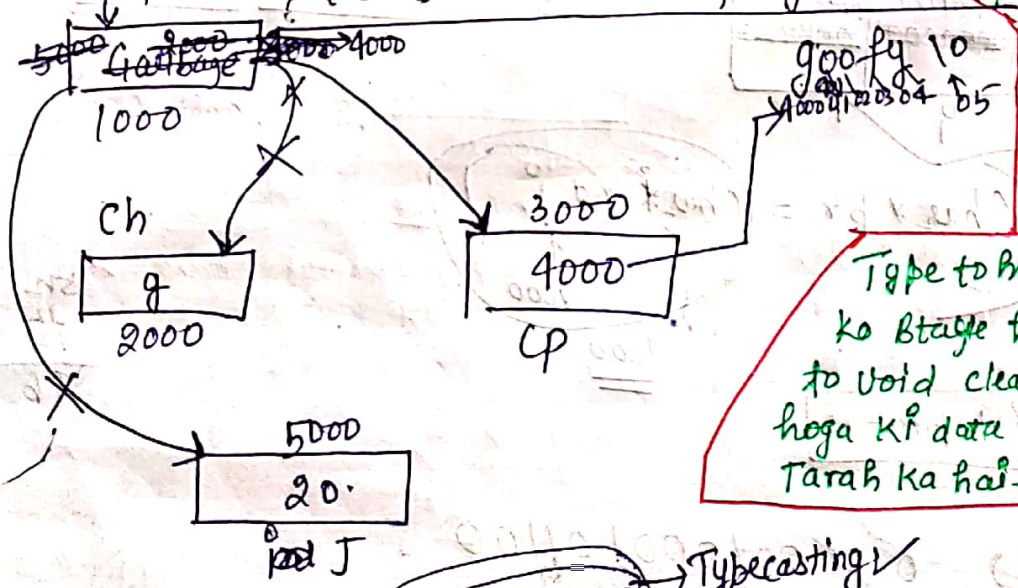
```
main ()
{
1) void *vp;
2) char ch = 'g';
3) char *cp = "goofy";
4) int J = 20;
5) vp = &ch
6) printf ("%c", (char*) vp);
7) vp = &J;
8) printf ("%d", *(int*) vp);
9) vp = cp;
10) printf ("%s", (char*) vp+3);
}
```

Note ★★★★★

- Agar hm void pointer ki type casting kr de to coo void pointer us tarah ke variable ko point to krega hi saath-2 usko hm access bhi kr skte hai. Lekin agar type casting nhi kiya to void pointer address to contain kr lega pr us address pr rakhe variable ko hm access nhi kr skte hain kyunki type to hm log void ko batate hi nhi to void clear hi to nhi hoga ki data type ki's tarah ka hai.

Sol^n

vp → vp tells I want address of anyone



Line → 6 ⇒ *{(char*) vp} → Typecasting into char type

⇓

g) O/P

Line (8) → {*((int*) vp)} → Typecast into integer type

vp → 5000

→ 20 → O/P ✓

Line-(10) → {((char*) vp+3)}

vp → 4000

→ g20fy\0

4000

↓

(fy) → Print ✓

O/P → g, 20, fy } ✓ → (g20fy) ✓✓

Que

```
int main()
{
1) char *str1 = "Hi";
2) char *str2 = "Bye";
3) fun1 (str1, str2);
4) Printf ("%s %s", str1, str2);
5) fun2 (&str1, &str2);
6) Pf ("%s %s", str1, str2);
}
```

What is the o/p?

```
Void fun1 (char *s1, char *s2)
{
  char *tmp;
  tmp = s1;
  s1 = s2;
  s2 = tmp;
}
```

double pointer

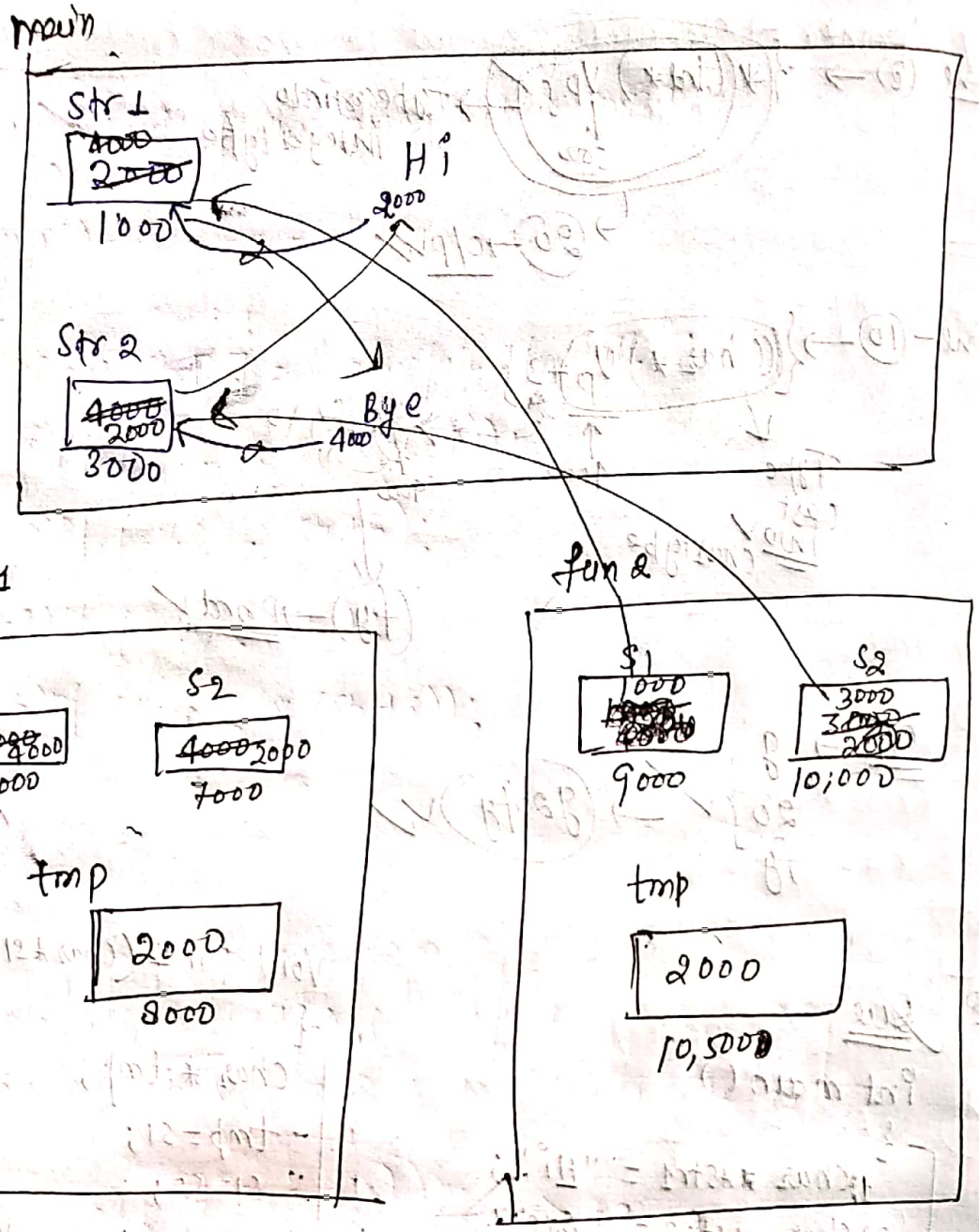
```
Void fun2 (char **s1, char **s2)
{
  char *tmp;
  tmp = *s1;
  *s1 = *s2;
  *s2 = *tmp;
}
```

Inside f^n only one pointer use because from Here we can manipulate only main f^n pointers. ✓

S1 & S2 stores Pointer of Pointers. {double Pointers}

Scanned with CamScanner

Sol^n



O/P Hi, Bye
~~2000~~, ~~4000~~, Bye, Hi

Blank & Null character Both are Not same
" " "\0"

Blank is one of the ASCII character
1B

Scanned with CamScanner

Que

```
void foo(char *a)
{
  1) if(*a && *a != ' ')
  {
    2) foo(a+1);
    3) putchar(*a);
  }
}
```

The O/p of the above function on input "ABCD EFGH".

Sol^n :- String Passing means first character Address Passing

Here I/p = "ABCD EFGH

01 02 03 04 05 06 07 08

1000

Pass where it call

AB CD EFGH

a 1000 2000

if ((*a) && (*a) != ' ')

1 4 2 3

A	A	1000
B	B	1001
C	C	
D	D	

Here
*> &&
Uniary
*> !=
!= > &&

foo(char *a) (a→1000)
1 True
2 foo(char *a) (a→1001)
1 True
3 2 foo(char *a) (a→1002)
1 True
3 2 foo(char *a) (a→1003)
1 True
3 2 foo(char *a) (a→1004)
False
3

O/P
DCBA

Que

Find Output of the given below program.

```
main()
{
  1) Char Str[ ] = " peacefull for all";
  2) Pf ("y.c", *(&Str[2]));
  3) Pf ("y.s", Str+5);
  4) Pf ("y.s", Str);
  5) Pf ("y.c", *(Str+2));
}
```

Sol^n

Str 1000	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17
p	e	a	c	e	f	u	l	l		f	o	r		a	l	l	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

2) * & Str[2] → a

a 1002

3) Str+5 1000 1005 → full for all

4) Str 1000 → peacefull for all.

5) * (Str+2) 1000 1002 → a

Q. main ()

```
{
  char a[5] = "hello";
  char b[5] = "hello";
  if (a == b)
  Pf ("hi");
  else
  Pf ("bi")
}
```

Sol^n

	1000	01	02	03	04	05
a	h	e	l	l	o	\0

	2000	01	02	03	04	05
b	h	e	l	l	o	\0

if (a == b) → falsy

1000   2000

D/P ⇒ "bi"

if at the place of a==b → 1000 a == b 2000

Then Error message will look that is L-Value can not be change { Base Add can not be change }.

if at " " a==b → *a == *b

Then *a == *b ⇒ hi will be printed

h   h

Scanned with CamScanner

Que

```
main ()
{
    char *a = "%d\n";
    a++;
    a++;
    pf (a-2, 300);
}
```

Solⁿ:-

"%d\n"

2000 2001 2002

a

2000 2001 2002

1000

a++;
a++;

pf (a-2 , 300);

Base Add of "%d\n"

pf (2000, 300);

|||

pf ("%d\n", 300);

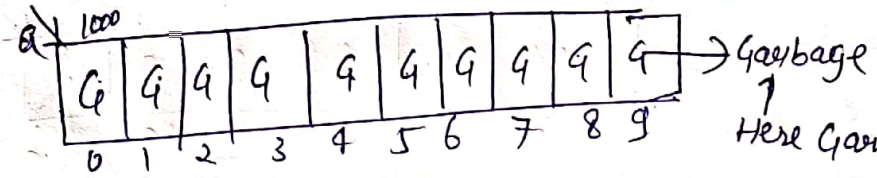
o/p → 300

Scanned with CamScanner

Que

```
main ()
{
1. char a[10];
2. char *b = "ghij";
3. size of (a,b);
4. size of (*a, *b);
5. int aa=10;
6. int *b = &a;
7. size of (b);
8. sizeof (*b);
}
```

Sol^n



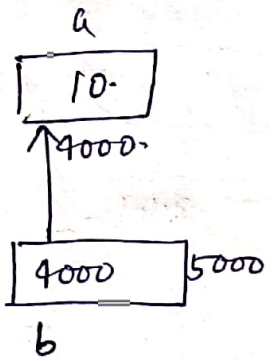
Here Garbage contain 1B Because of Char type

size (a, b) → 10, 8 (10B, 8B) — Address storing

size (*a, *b) → 1, 1 (1B, 1B)

size (b) → 8B → 8

size (*b) → 2B ✓ → 2



O/P , 10, 8, 1, 1, 8, 2 ✓

Scanned with CamScanner