CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM

Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

PROGRAMMING LANG.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)2

PROGRAMMING LANGUAGE

TOPICS :- programming in C : Functions

* * pointers
strings
Recursion
parameters
passing Techniques

* * * Static Scoping and
Dynamic Scoping
Binding
- Storage class.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

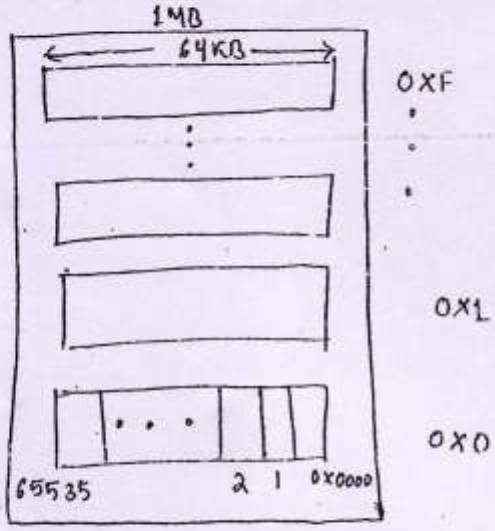
memory is collection of segments, each segment is collection of bytes (one segment = 64KB) on 16 bit compiler)

→ Every Byte identified by a unique location number called address.

→ To store address of a Byte we need pointer variable.

→ when a program is loaded to the memory it occupies 1MB of memory. This 1MB will be decide into segments where segment address are stated will 0X0000 and ending with 0XFFFF.

→ Each segment is of size 64KB, so their addresses are starting from 0X0 and ending with 0XF i.e. in decimal form it can be represented as starting byte address is zero and last byte address is 65535.



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
Main( )
{
int i = 10;
int *p;
p = &i;
printf ("%d %d", i, *p);
printf ("%u %u", &i, p);
printf ("%d", p);
printf ("%p %p", &i, p);
printf ("%u", &p);
*p = 50;
printf ("%d %d", i, *p);
}
```

i: $\boxed{p = i}$ ← error.

p → 265 265

printf("%d", p); — 504 (unknown)

printf("%p %p", &i, p); (hexadecimal of 200)

printf("%u", &p); → 300

*p = 50; → 50 50

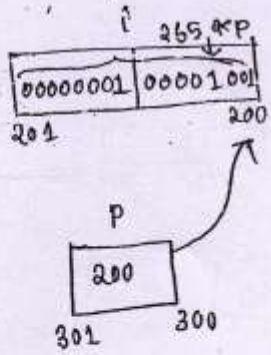
i: $\boxed{265}$ 50 ← *p.

→ %p, %x printf address of 16 bit OS in the form of hexadecimal notation.

→ %u print address of 16-bit OS in the form of decimal notation.

→ %lp, %lx, %lu print address of 32-bit OS.

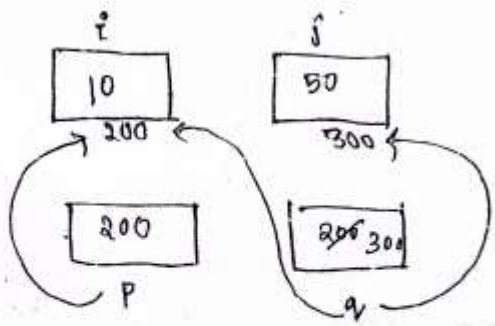
→ If p is a pointer to print result of p, use %u formats specification don't use %d.



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

POINTER ASSIGNMENT

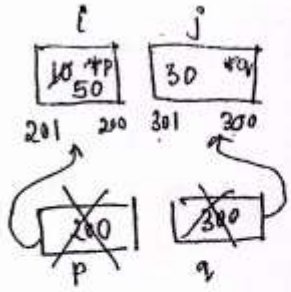
```
* main ( ) {
    int i=10 , j=50;
    int *p , *q;
    p= &i ; q=p;  → pointer Assignment.
    printf ("%d %d %d", i, *p, *q);   10 10 10
    q = &j;
    printf ("%d %d %d %d", i, *p, *q, j);   10 10 50 50
  }
```



Passing Address to a function :-

```
change (int * , int * );
main( ) {
int i =10 , j=50 ;
change (&i ,&j);
printf ("%d %d", i ,j);
}
change ( int *p , int *q)
{
*p = 50 ;
*q=30 ;
}
```

(200, 300)

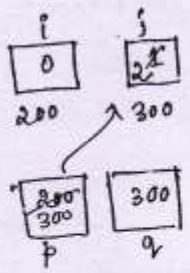


CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
int y( int *p , int *q)
 {
 p=q;
 *p=2;
 }
int i=0 ; j=1 ;
main()
 {
 y( &i, &j);
printf ("%d %d", i,j);
 }
```

when localvariable is not available then searching and passing global variable.

O/P → (0,2)



→ O/P → (0,2)

How much memory a pointer can occupies :-

There are 2 types of pointer variable.

1. Near
2. Far.

Near pointer :-

→ It works with only one segment.

→ If we increase the near ~~segment~~ pointer it points to next OFFSET address.

→ when the entire program fix within one segment then to store address of any variable of that program we need near pointer and it occupies 2 bytes.

→ By default every pointer is near pointer.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

FAR POINTER :-

→ If the entire program doesnot get within one segment then to store address of any variable of that program we need far pointer.

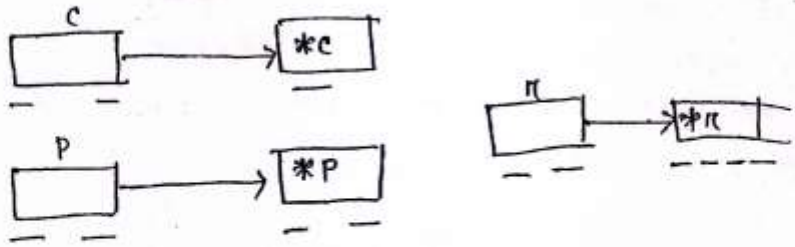
→ It occupies 4 bytes (2 bytes for segment address and 2 bytes for offset address).

→ It can handle more than one segment but it can work with only one segment at a time.

→ If we increase far pointers, it points to next offset address only.

How much memory a pointer can Access:-

```
main( )
{
char *c;
int *p;
float *r;
printf ("%d", ~~size of~~ sizeof(c));      2 bytes
printf ("%d", sizeof(*c));                 1 byte
printf ("%d", sizeof(p));                  2 bytes
printf ("%d", sizeof(*p));                 2 bytes
printf ("%d", sizeof(r));                  2 bytes
printf ("%d", sizeof(*r));                 4 bytes
}
```



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

→ **sizeof** is a operator but not a function which return no of bytes occupied by the argument.

→ Accessing memory by the pointer depends on the data type of the pointer, i.e., if

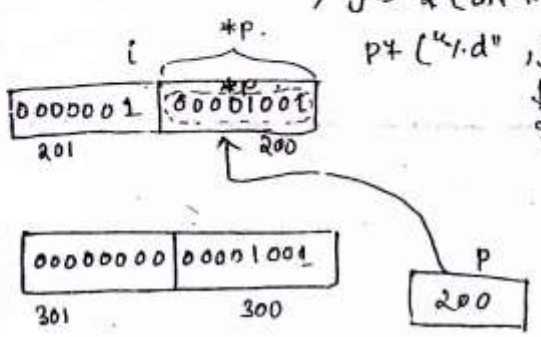
int *p → then p can access 2 bytes of memory.

POINTER CONVERSION :-

```
main()
{int i=265, j;
 char *p;
 p=&i;
 j=*p;
 printf("%d", j);
}
```

char ptr. contains int addr. not given any error and accept but given warning of "suspecious pointer conversion"

→ j = *(int *p);
printf("%d", j);
↓
265



→ The above program generate warning message like "suspicious pointer conversion" coz we are storing address of integer into char. pointer, this assignment is allowed but while accessing i, using p the expr. *p can access only one byte, to overcome this convert p as integer pointer, temporarily using the statement → $\boxed{j = *(int\ *)p}$

now, j holds the value 265.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Pointer initialization:-

```
main()
{
int *p;
printf("%d", *p);
}
```

→ The pointer is not initialized so, it contain garbage Addr, the expression *p occurs the value from the memory which is not result for our program. It leads illegal access, system may crash.

NULL POINTER :-

```
main()
{
int *p = NULL;
*p = 30;
printf("%d", *p);
}
```

a) 30 b) NULL c) error d) none of there

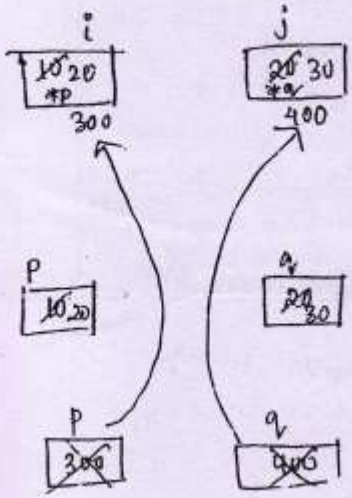
→ The above program return error - "NULL POINTER" "assignment at runtime", coz we are storing 30 in no address location.

call by value & call by reference:-

```
main()
{
int i=10, j=20;
printf("%d %d", i,j);      → (10, 20)
call1 (i, j);
printf("%d %d", i,j);      → (10, 20)
call2 (&i, &j);
printf("%d %d", i,j);      → (20, 30)
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS) 10


```
call 2 ( int p, int q)
  {
  p = p+10 ;
  q = q+10 ;
  printf ("%d %d", p, q) ; → (20,30)
  }
call2 (int *p , int *q):
  {
  *p = *p+10 ;          → 20
  *q = *q+10 ;
  printf ("%d %d", *p, *q); → (20,30)
  }
```



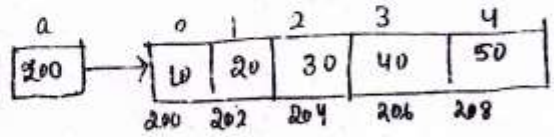
(i, j) = (20, 30)

(p, q) → (20, 30)

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

POINTERS TO ARRAY

Let us consider int a[5]= {10, 20, 30, 40, 50};



① Base address of Array can be represented in two ways.
- i.e. a or &a[0]
  a → 200, &a[0] → 200

② Array name itself a constant pointer i.e. we cannot increment or decrement the constant pointer i.e.

a++, a--, ← invalid statements.

It return error is "L-value required".

Ex:- a++
a = a+1
a = (a+1×2)
a = 202 ← not base addr. so gives error.

③ Accessing mechanism of array elements using array name.

Ex:- a[0] = *(a+0×size of data)
= *(200+0×2)
= *(200)
= 10

a[2] = *(a+2×2)
= *(200+4)
= *(204)
= 30

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

$\boxed{a[i] = *(a + i * \text{size of data})}$

12

Q:- How to make a pointer to the array?

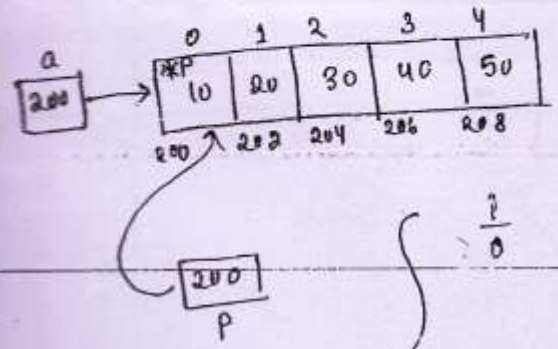
Q:- How to Access Array elements using pointer?

```
main()
{
int a[5] = {10,20,30,40,50};
int *p;
```

'M' // p=a; (or) p= &a[0] // making pointer to array.

```
for (int i=0; i<5; i++)
{
printf("%d", p[i]);   // First way of accessing array elements using pointer.
}
}
```

O/p → 10, 20, 30, 40, 50



i	p[i]
0	p[0]
	*p(200+0×2)
	*p(200) = 10
1 →	20
2 →	30
3 →	40
4 →	50

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Second way of accessing array elements using pointers.

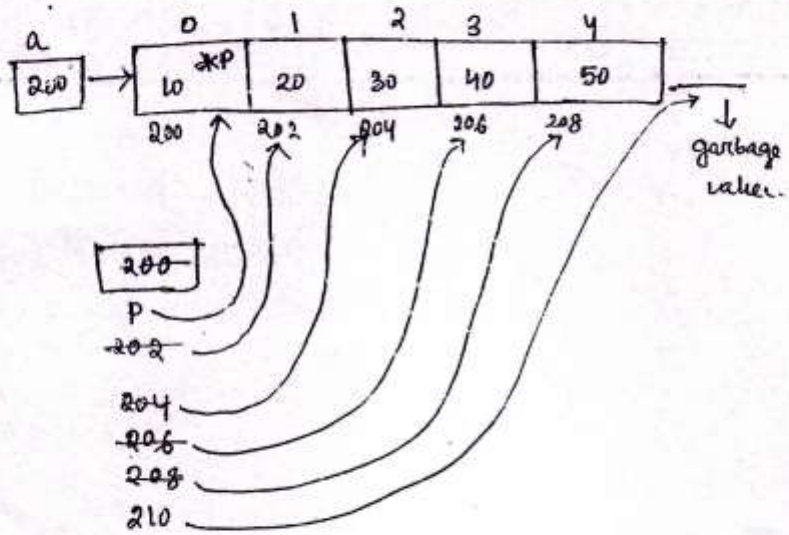
→ Replace eqn (*) by the statement :-

```
printf ("%d", *p);
p++;
}
```

O/p

i	*p	p++
0	10	202
1	20	204
2	30	206
3	40	208
4	50	210

int p	int *p	float *r
p = 200	p = 200	r = 200
p = p+1;	p = p+1;	r = r+1
= 200+1;	= 200+1×2	= 200 × (1+4)
p = 201	p = 202	r = 204



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS) 14

Third way of Accessing Array elements using pointers. (+)

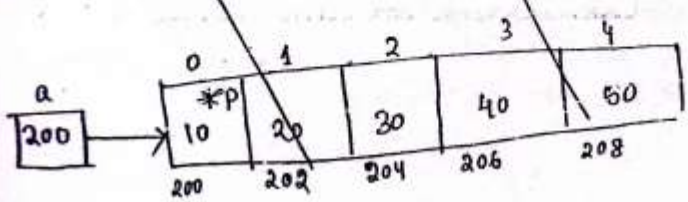
Replace eqn (*) with statement :-

```
printf("%d", *(p+i));
}
```

o/p

i	*(p+i)
0	*(200 + 0×2) = *(200) = 10
1	20
2	30
3	40
4	50

int p	int *p	float *r
p = 200;	p = 800;	r = 200
p = p+1 = 200+1	p = p+1 = 200 + (1×2)	r = r+1; = 200 + (1×4)
p = 201	p = 202	r = 204



Passing 1-D Array as an argument to the Function.

```
main ()
{
int a[5] = {10, 20, 30, 40, 50};
display(a, 5);
printf("%d", sizeof(a));
}
```

→ 10 bytes.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
display ( int *p, int n )
{
for ( int i=0; i<n ; i++)
{
printf("%d", p[i]);
printf("%d", sizeof (p));
}}
```

10, 20, 30, 40, 50

→ 2 bytes.

In C, there is no provision to pass all array elements in a single shot to called function.
If we assume, it is possible then we might think that it is required to declare another array of same size in the called function. It leads 2 problems

1. memory wastage.
2. Redudant copy of data.

To overcome this pass base address of array and make a pointer to that base address. In the above problem array size is 10 bytes, we are accessing these 10 bytes using a pointer of size 2 byte. It is an advantage of making a pointer to the array.

Gate 2010 :-

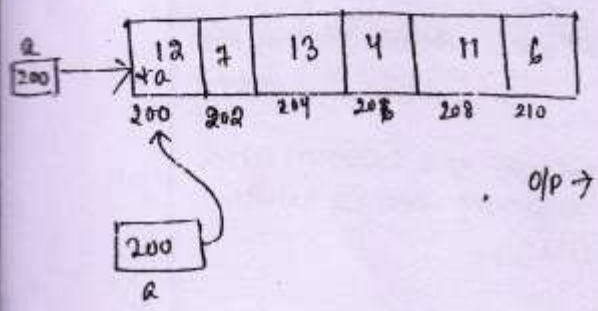
```
int f ( int *a , int n)
{
if (n≤0)
return 0;
else
if (*a % 2 = = 0 )
return *a + f(a+1, n-1);
else return *a - f(a+1, n-1);
}
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

(8)

16

```
main( )
{
int a[ ] = { 12, 7, 13, 4, 11, 6 };
printf ("%d", *(a+6));
}
```



O/P → 15 .

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

POINTER TO FUNCTION

Q:- How to declare a pointer to a Function ?
Q:- How to make a pointer to a Function ?
Q:- How to call a function using pointer ?

Syntax to declare a pointer to function:-

Returntype of Function (*pointer variable)(List of arguments);

Example-1

void (*p)( ); . p is a pointer to a function which takes no argument and its return type is "void" ;

Example-2

int(*p)(int, int) ; // p is a pointer to a function which takes 2 integers as an argument and its returntype is "integer"

Example-3

int (*f)(int *) ;

f is a pointer to a function which takes integer pointer as an argument and its return type is integer.

How to make a pointer to a function:-

Syntax:- pointer variable = function name without parenthesis

Ex:-1 p = test

```
void test ( )
{
printf ("Hello");
}
```

Syntax to call a function using pointer:-

pointervariable (List of arguments);

OR

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS) 18

(* pointer variable) (list of arguments);

```
void test ();
void main ()
{
void (*P)( ); // Declaration
p=test ;      // making of pointer to function
P( );         // calling the function.
(*P)( );
}
```

```
void test ( )
{
printf ("Hello");
}
```

printing 'Hello' Two times.

```
int get( )
void put (int);
main( )
{int K;
D // int (*P)( ) ;
void (*q) (int);
4 // p=get ;
q=put ;
C // K=P( );
q(K);
}
```

```
int get ( )
{
return 10;
}
void put (int l)
{
printf ("%d", l);
}
```

```
* main( )
{
int n;
void (*p)( ) ;
printf ("enter your choice [1 2 3] ");
scanf ("%d", &n);
switch( n )
{
case 1 :  p=x ;
          break;
```

```
void x( )
{
printf ("we called x");
{
void Y( )
{
printf ("we called Y");
}
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
case 2:  p=Y;
         break;
case 3 :  p=Z;
         break;
Default:  pf("invalid choice");
          return;
    }
  p( );
}
```

Passing address of function as an argument to another function:-

→ when a function is loaded it is stored in separate physical memory location. All instruction of the function converted into machine code form and entry point of the machine code address is stored into the function name.

→ If we make a pointer to the function the base address of starting instruction is stored into the pointer, using this control is shifting from calling function to called function.

→ By Default all functions are global in 'c' so we can call them from wherever we want. So there is no such significance to make a pointer to functions.

```
void X( );
void Y( );
main( )
{
D// void (*p)( );
M// p=X;
    Y(p);
    p( );
}
```

```
void X( )
{
pf("we called x");
}
```

```
Y(void (*q)( ))
{
q( );
}
```

but we cannot declare p( ) here.

we can access x( ) in by two ways.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

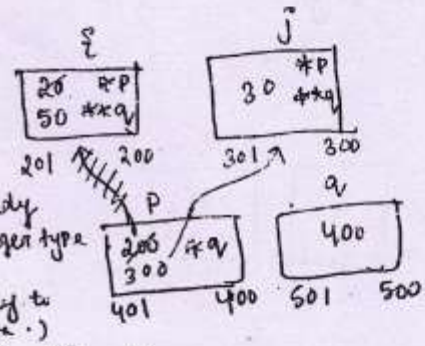
Pointer - to - pointer :-

(20)

Note:-

int i ; datatype of i. is integer
int *p ; " " p is int*
int **q ; " " q is int**

example-1




```
main( )
{
int i=20; int j=30;
int *p;      (p is a pointer ready to receive integer type data)
int **q;     (q is a pointer ready to receive int** type data.)
p= &i;
q= &p;
printf("%d %d %d", i, *p, **q);   // 20 20 20
**q = 50;
printf("%d %d %d", i, *p, **q);   // 50 50 50
*q= &j;
printf("%d %d %d %d", i, j, *p, **q);   // 50 30 30 30
```

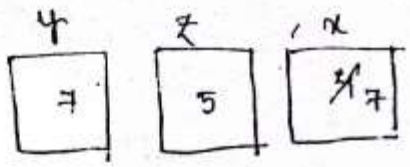
Note-2003

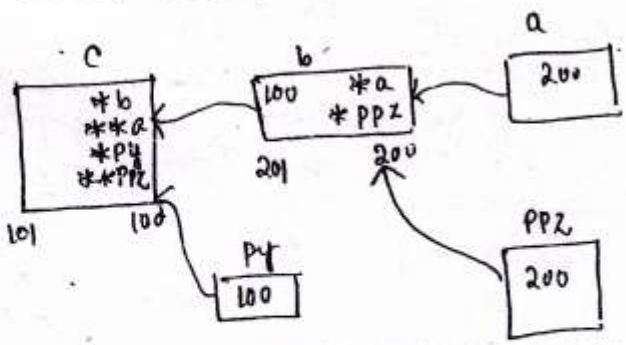
```
void main ( )
{
int c, *b, **a;
c=4; b= &c; a= &b;
printf("%d", f(c,b,a));
}
~~printf(in~~
int f( int x, int *py, int **ppz)
{
int y, z;
**ppz = **ppz+1;   ⟶ 5
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

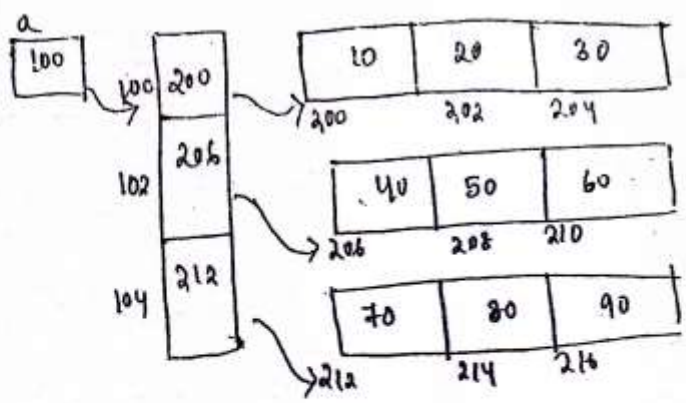

```
z = **ppz
*py = *py + 2 ;
y = *py ;          → y = 7
x = x + 3 ;        → x = 7
return x+y+z ;     → 19
}
```

(i) 19
(ii) 90
(iii) 21
(iv) 22





NOTE:-
good example for pointer to pointer 2 dimensional array
int a[3][3] = {10, 20, 30, 40, 50, 60, 70, 80, 90}



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)22

Accessing mechanism :-

a[0][0] = *(a + 0 × size of ptr) [0]
= *(100 + 0×2) [0]
= *(100) [0]
= 200 [0]
= *(200 + 0 × size of data)
= *(200 + 0×2)
= *(200) = 10

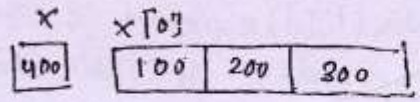
a[i][j] = *(*(a + i × size of ptr) + j × size of data)

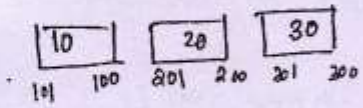
NOTE :-
If int a [i][j] represent 2 dimensional array then
(i) a[1][2] = 150 ; is valid statement and it replaces the value of 1st row 2nd column element with 150.
(ii) a[i] represent base address of ith row. It is allocated by the compiler at runtime so we cannot change it. i.e. a[i] = 1000; invalid statem

Array of pointers :-
(1) float a[20] ; // a is an array of '20' float values and its size is 80 bytes.
(2) float *a[20] ; // a is array of 20 pointers which can receive address of float variables. and size of 'a' is 40 bytes.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
main ( )
 {
int a=10, b=20, c=30;
int *x[3];      ( x is array of 3 pointer which are
                  ready to receive integer).

x[0]= &a;
x[1]= &b;
x[2]= &c;

for (int i=0; i<3; i++)
  {
  printf("%d", *x[i]);
  }
```





i	*x[i]
0	*x[0] = (*(*(400+0×size of ptr)))
	= *(*(400))
	= *(100)
	= 10
1	*x[1] = 20
2	*x[2] = 30

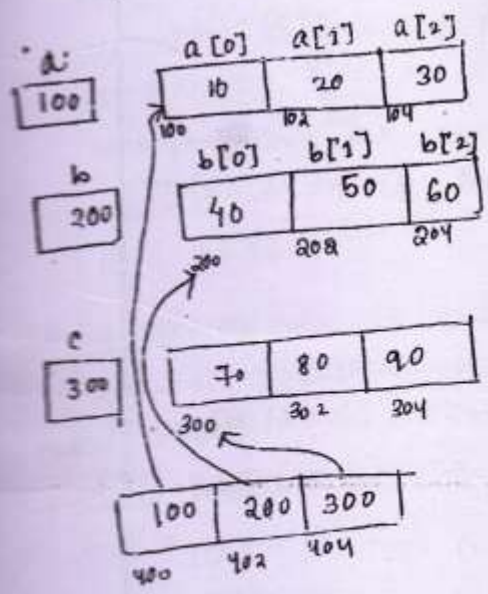
Q:- main ( )

```
 {
int a[3] = {10, 20, 30};
int b[3] = {40, 50, 60};
int c[3] = {70, 80, 90};
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

int *x[3]

x[0]=a;
x[1]=b;
x[2]=c;
for (int i=0;i<3;i++)
{
printf ("%d %d %d", *(x[0]+i), *(x[1]+i), *(x[2]+i));
}



O/P:- 10, 40, 70, 20, 50, 80, 30, 60, 90.

i	*(x[0]+i)	*(x[1]+i)	*(x[2]+i)
0	*(100+0×2) (size of data)	*(200+0×2)	*(300+0×2)
	*(100)	*(200)	*(300)
	10	40	70
1	20	50	80
2	30	60	90

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

NOTE:- (i) In the above example elements of array a, b, c can be represented by using 'x' as follows.

a[0] = 10	b[0] = 40	c[0] = 70
x[0][0] = 10	x[1][0] = 40	x[2][0] = 70
a[1] = 20	b[1] = 50	c[1] = 80
x[0][1] = 20	x[1][1] = 50	x[2][1] = 80
a[2] = 30	b[2] = 60	c[2] = 90
x[0][2] = 30	x[1][2] = 60	x[2][2] = 90

(ii) In the above example if we use the statement x[2][1] = 35. Then 2nd element of 'c' is replaced by 35 i.e. 80 replaced by 35.

Gate 2023

Consider the following declaration int *A[10];
int B[20][20];

Of the following expressions (i) A[2] (ii) A[2][3] (iii) B[2] (iv) B[5][5]

It will give error. ← (iii)

which one will not give compile time error if we use them to left side of assignment statement in 'C'

(a) I, II, IV
(b) IV only
(c) I, III, IV
(d) II, III, IV

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)26

POINTER TO STRING:-

In C there is no special datatype to declare string variable. so we are taking help of character array.

String:-

syntax:- char arrayname [size];

ex:- char Sname [50];

In 'C' some functions like printf, strlen( ) might not aware the end of the string. to overcome this all standard library functions use Null characters. string terminator.

→ we can also use 0 as string terminator becoz it is ASCII value of '\0'

String constant:-

group of characters enclosed in double quotes is called string constant.

→ all characters of string constant stored in the memory sequentially in contigeous memory location.

→ string constant automatically return base address of the string.

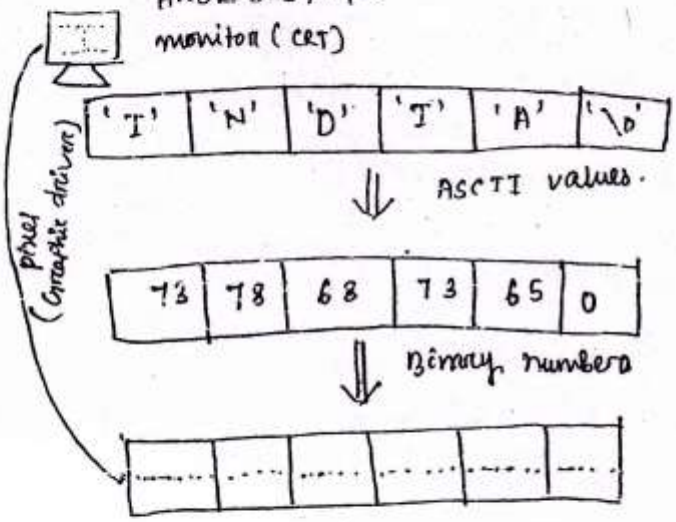
→ whatever operation performed on array can also perform on string constant.

→ every character of string constant is stored in the form of character constant and this will be converted into corresponding ASCII values. since we are working with digital computer it can understand only binary numbers so. These ASCII values convert into corresponding binary ~~values~~ numbers.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Ex:- "INDIA"
"apple"
"0402717032"
"HNO_ 323, HYD"

I → character
'I' → character constant




```
main()
{
if ("sai" == "sai")
printf ("strings are equal");
else
printf ("strings are not equal");
}
```

In the above program base address of 2 string get compare, since no two strings have same base address so else part will be executed i.e. strings are not equal.

```
main()
{
if (*"sai" == *"saibaba")
printf ("string are equal");
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
else
  printf("          ");
}
```

*300 = 'S'
*400 = 's'

In the above program 1st character of 2 strings get compared since they are equal if-condition will executed.

Intialisation of String variable:-

```
char s[] = "INDIA";
char s[6] = "INDIA";
char s[] = {'I','N','D','I','A','\0'};
char s[6] = {'I','N','D','I','A','\0'};
char s[] = {73, 78, 68, 73, 65, 0};
char s[6] = {73, 78, 68, 73, 65, 0};
```

donot write ASCII value '73' ✗

```
char p[20];
char *s = "string";
int length = strlen(s);
for (int i=0; i<=length; i++)
    p[i] = s[length-i];
printf("%s", p);
```

for '%s' always pass address, 'p' hold address so 'p' whenever compiler get null character it observe that no string.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

(a) gnirts
(b) nirts
(c) string
✓ (d) no output printed

0	1	2	3	4	5	6
's'	't'	'r'	'i'	'n'	'g'	'\0'
100	101	102	103	104	105	106

s: 100 → 100

p: 200 → | '\0' | 'g' | 'n' | 'i' | 'r' | 't' | 's' | '\0' |

by reading null character compiler observe no string is there
o/p → no o/p printed.

```
char *p = "GATE2011";
printf("%s", p+p[3]-p[1]);
```

(1) GATE2011
(2) E2011
✓ (3) 2011
(4) 011

$100 + E - A$
$= 100 + 69 - 65$
$= 104$

p: 200

p[0]	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	p[7]	p[8]
'G'	'A'	'T'	'E'	'2'	'0'	'1'	'1'	'\0'
200	201	202	203	204				

$200 + (69 \times 1 - 65 \times 1)$
$= 200 + 4 = 204$

69 → scalar
200 → address

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

whenever adding scalar to address multiply the scalar with size of data. 15 (30)

```
printf("%c", p+p[3]-p[2]);
```

→ unknown value we cannot access address by using '%c' so unknown value.

```
printf("%c", *(p+p[3]-p[1]));
```

*(add) → value.

* give o/p → '2'

Q. main()

```
{
char *p = "haiFriend";
char * p1;
p1=p;
while(*p != '\0')
{
*p++;          ⟶  ++*p;    ++*p++;
printf("%c", *p);
}
printf("%s %s", p, p1);
}
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Q:- main ()

```
{
int i=3;
printf("%d %d", i, i--)
}
```

printf statement

L → R

operation perform

L ← R

o/p → 2,3.

```
main ()
{
int i=4, j;
j = (++i) * (++i) * (++i)
```

↓ 7 ↓ 6 ↓ 5

printf → 7*6*5 = 210

UPTE 2000

```
abc (char *s)
{
if (s[0] == '\0')
return;
else
  abc (s+1);
  abc(s+1);
printf ("%c", s[0]);
}
```

```
main ()
{
abc("123");
}
```

Q:- what is o/p ? 3323321.

Q:- If the string has 'n' character then
(a) how many characters to printed ?
(b) How many function calls are there?

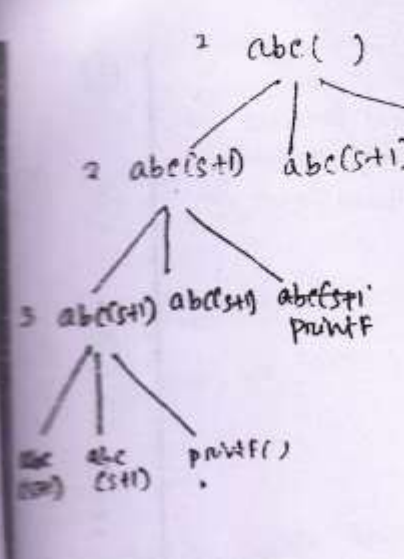
3.

abc (s+1)

5

5

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)



no of printf		no of Function calls
0		1 $= 2^0$
1	$= 2^0$	2 $= 2^1$
	+	+
2	$= 2^1$	4 $= 2^2$
	+	+
4	$= 2^2$	8 $= 2^3$
$n=3$	$2^0+2^1+2^2 \ldots$	$2^0+2^1+2^2+2^3$
$n=n$	$\underbrace{2^0+2^1+\cdots 2^{n-1}}_{n \text{ term}}$	$\underbrace{2^0+\cdots 2^n}_{n+1 \text{ term}}$

Storage classes :-

In 'c' every variable posseses some characteristic like data type scope and lifetime.

→ storage classes describe scope and lifetime of a variable.

→ c support 4 types of storage classes.

(i) Automatic
(ii) Static
(iii) register
(iv) external.

Scope :- It is the region in which a variable is available and accessable.

Lifetime :- It explains that when the variable comes into existance and when the variable goes up.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
int i=100;        →   Global variable.
main()
{
int i=10, j=15;
  {
  int i=5;                  → 15
    printf("%d", i, j);     → 15
  }
printf("%d %d", i, j);      → 10
                            → 15
}
```

Local variable have high priority than global variable

Global variable is visible to everyblock but there should not be any local variable with same name.

```
int i=100;
main()
{
int i=100.
  {
  int i=10, j=50;
  printf("%d %d", i, j);    → 10, → 50
  }
printf("%d %d", i, j);      → (error) undefined symbol.
                            → 100
}
```

The second printf statement called undefined symbol 'j'.

Automatic :-

```
void test();
main()
{
test();
test();
test();
printf("%d", k);
}
```

error undefined symbol.

```
void test
{
auto int k=10;
printf("%d", k);
k++;
}
```

→ local scope.

1st 2nd 3rd

201 200 201 200 201

10 10 10

10 10 10

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

(1) auto datatype variable name;
ex: auto int i;

(2) By default every variable is auto variable
(3) an autovariable is not initialize it contain the garbage value.

(3) It is created for every function call and Reinitialized
(4) It has local scope and local lifetime.

Static :-

(1) syntax :- static datatype variable name;
ex:- static int i;

(2) if static variable is not initialized then it contains the value '0'.

(3) It is created only once and persisting previous stage values from the distruction of several function call.

(4) It has local scope and program lifetime.

```
void test( )
main( )
{
test( );
test( );
test( );
printf("%d",K)
}
```

undefined symbol (local scope)

```
void test( )
{
static int K=10
printf("%d",K);
K++;
}
```

K: ~~10~~ ~~11~~ ~~12~~ 13

10
11
12

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Date 2004

```
int f(int n)
{
static int i=1;
if (n>=5)
   return n;
   n=n+i;
i++;
return f(n);
}
```

1

what is return value of f(1).

Ans:- 7.

Date 2000

```
int incr(int i)
{
static int count=0
   count = count+i;
   return (count);
}
main()
{
int i,j;
for (i=0; i<=4; i++)
    j= incr(i);
}
```

count: ~~0~~ ~~0~~ ~~1~~ ~~3~~ ~~6~~ 10

i: 0

j: ~~1~~ ~~0~~ ~~3~~ ~~6~~ 10 → j: 10

what is the value of 'y' at end of execution?

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

r: 5

f(3)+2
f(2)+5+2
f(1)+5+5+2
f(0)+5+5+5+2
1+5+5+5+2
= 10

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)36

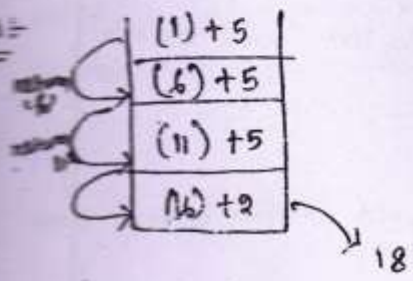
f(n-1)+n ;
}

18

what is the value of f(5)?

An

ans:-

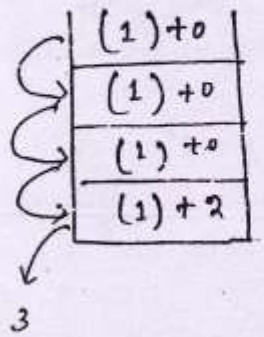


Q

Q:- In the above program if line-1 is replaced by auto int n=0, Then what is the return value of f(5).

ans:-3

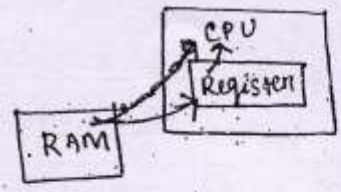
f(3) + 2
↓
f(2)+0 + 2
↓
f(1)+0 + 0 + 2
↓
f(0) + 0 + 0 + 0 + 2
↓
1 + 2 = 3



Reg

Register :-

```
main ( )
{
register int i=10;
printf ("%d", i); //(i) will be accessed faster.
}
```



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

The Keyword 'register' is not a command it is just request to the compiler to allocate memory into register ~~stead~~ instead of RAM. If ~~fires~~ free registers are available. Then memory allocated into registers. otherwise memory allocated in RAM only.

→ By default register variable behave like "auto variable"

Gate 2012

```
int a, b, c=0;
void prtfun( );
void main( )
{
line① static int a=1;
prtfun( );
a= a+1;
prtfun( );
printf ("%d %d", a, b);
```

a b c
[0] [0] [ ]
garbage value

a
[1]
100

a
[2]
200

```
void prtfun ( )
{
static int a=2; — line (2)
int b=1;
a= ++b;
printf ("%d %d", a, b);
}
```

a → 4, b → 2

a = a+(++b)

o/p → 4 2
6 2
2 0

1) what is o/p ?
2) If Line 1 replace by auto int a=1;
line 2 replace by register int a=2;

ANS:- then o/p is → ~~4 2~~ 4 2
4 2
2 0

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)38

External (Global Variable)

```
int a = 10;
main ()
{
printf ("%d", a);   → 10
change ();
printf ("%d", a);   → 20
}
void change ()
{
a=20;
printf ("%d", a);   → 20
}
```

a: 10 → 20

O/P → 10 20 20

OH we will declare any datatype then it is global variable.

In the above program main() and change() both are using global variable 'a' if global variable is not initialized then it contain the value '0'.

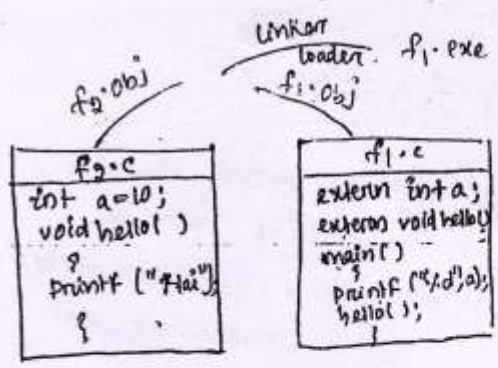
Extern (prototype):-

When the program size is very big then the code may be distributed into several files and these files compile independently and generate the object code. In this compilation process if one file is using global variable but it is declared in some other file then we get undefined symbol error. To overcome this we use extern keyword before using the global variable.

file.1 — f1.c
```
int a = 10;
```

f2 — f2.c
```
extern int a;
main()
{
printf ("%d",a);
change ();
}
```

f3 — f3.c
```
extern int a;
void change ()
{
a = 20
printf ("%d", a);}}
```



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

PARAMETER PASSING TECHNIQUES

(1) call by values
(2) call by References
(3) call by value Result (or) copy restore.
(4) call by Name.
(5) call by Text
(6) call by Need.

call by value :-

```
void swap (int a, int b)
{
int t;
t=a;
a=b;
b=t;
}
main ()
{
int K=50, m=25;
swap(K,m);
print (K,m);
}
```

print (K, m); → 50, 25.

t	a	b
50	~~50~~ 25	~~25~~ 50

K	m
60	25

· changes made in called function may or may not effect in the calling function.
If global variable then → effect the calling funct^n
otherwise → not " "

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)40


```
void swap( int a , int b)
 {
 int t;
 t=a;
 a=b;
 b=t;
 }
main( )
 {
 K=50, m=25;
 swap ( K , m) ;
 print (K, m);
 }
```

(100, 200)

t	a	b
50	100	200

K	m
~~50~~ 25	~~25~~ 50
101 100	201 200

25 50

In call by reference mechanism changes made in called function effect in the calling function.

Call by value Result (OR) Copy Restore :-

It is the combination of call by value and call by Reference.

```
void swap (int a, int b)
 {
 int t;
 t=a;
 a=b;
 b=t;
 }
main ( )
 {
 int K=50, m=25;
```

copy the

(100, 200)

t	a	b
50	25	50

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
swap(k,m)
print(k,m);
}
```

pa
Q:-

Note

Is call by value result and call by reference prints the same result always?

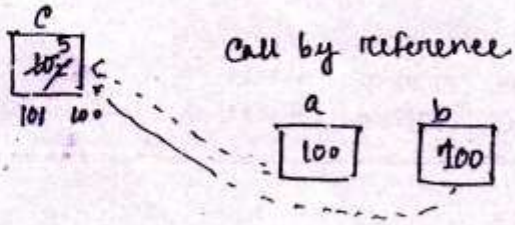
```
main()
{
int c=10;
test(c,c);
print(c);
}
```

(100, 100) → 5

```
void test(int a, int b)
{
a=2;
b=b+3;
}
```

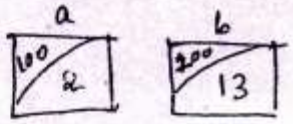
Pag
Q: 37

Call by reference.

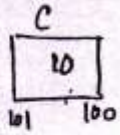


O/P → 5

Call by value.



O/P → 2 or 13



Ans:- no it's not printing the same result always.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

(42)

Q:

(5, 10, 10)

y = 14

z = 3 + 14 + 10 = 27

x: 10 (100) y: 3 (200)

x: 200 y: 100 z: 100

14 + 3 + 14

⇒ 28 + 3 = 31

o/p → (31, 3) (call by reference)

call by value, o/p → (10, 3)

Q. 37

i: 50 100

call by value..

o/p → 100, 60

'i' is global.

call by reference :-

i: 50 10 j: 60

x: 10 → 10 y: 70 → 70

o/p → 10, 70.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

314

```
procedure p(a,b,c)
 {
 a=2
 c=a+b;
 }
main ( )
 {
 x=1;
 y=5;
 z=100,
 p(x, x+y, z);
 write (x,z);
 }
```

(100, 200, 300)

call by value.

1, 100.

call by reference.

x	x+y	z	y
~~1~~ 2	5	~~100~~ 7	5
100	200	300	400

100	200	300
a	b	c

o/p → 2, 7

call

316 / 10

315 / 19

```
procedure p(x,y,z)
 {
 y = y+1;
 z=x+x;
 }
main( )
 {
 a=2;
 b=3;
 p( a+b, a, a)
 print (a);
 }
```

call by value.

o/p → 2

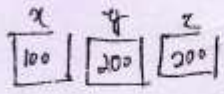
call by refer

a+b	a	b
5	~~2~~ 6	3
100	200	300

call

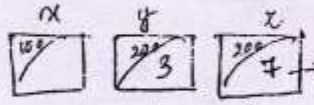
CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

O/P → '8'



44

call by value Result :-

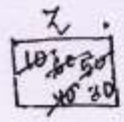


O/P → 3 or 7.

'3 or 7'
'a' having 200 location.

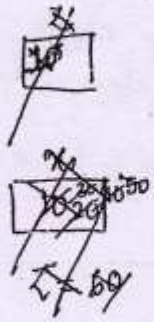
call

316/10

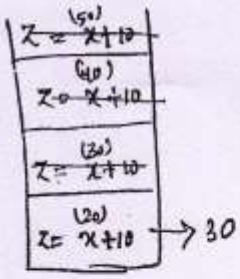



```
z ; integer :
recur ( x : integer)
  {
  if x <= 90 then
    {
    x := x+z;
    recur (x);
    z := x+10;
    }
  }
main ( )
  {
  z := 10;
  recur (z);
  write (z);
  }
```

20
30
40
50



~~References~~



call by value

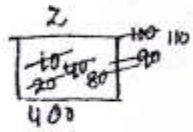
O/p :- 30.

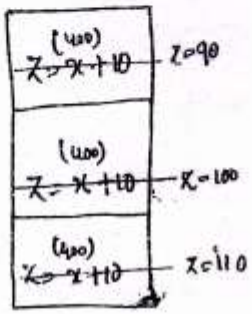
no of times recur ( ) is call by 'a'. value mechanism is.

call "a"

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Call by reference :-





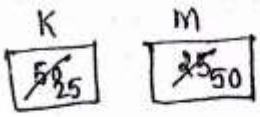
o/p → 110.

Call by Name :-

In call by name formal arguments renamed or replaced by actual arguments.

```
void swap ( int a, int b)
  {                  a → k,  b → m
  int t;
   t = a;            t = k;
   a = b;            k = m;
   b = t;            m = t;
  }

void main( )
  {
  int k = 50 ;
  int m = 25 ;
   swap(k, m);
   print (k, m);
  }
```

print(k, m) → 25, 50



CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)46

Q:- Is call by reference and call by name print the same result always?

```
main ( )
{
int i = 5 , A [50];
  A[5] = 10 ;
Swap ( i, A [i] );
Print ( i, A [5] );
}
```

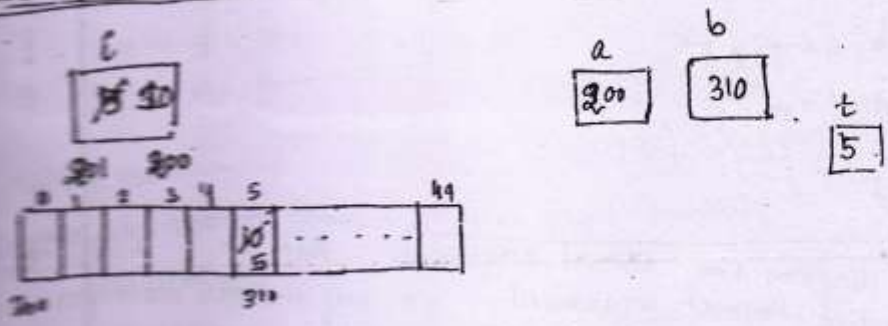
```
swap ( int a , int b )
{
int t;
t = a;
a = b;
b = t;
}
```

a → i (10), b → A[5] (5)

Print → i → 10, A[5] → 10

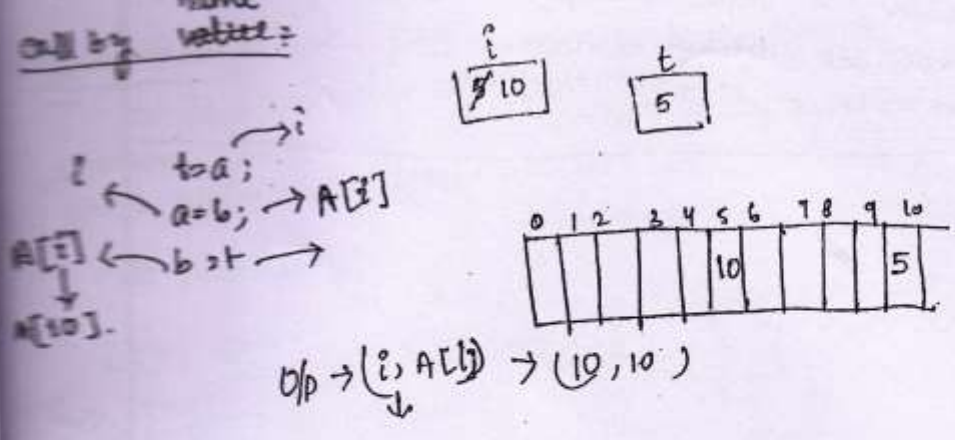
call by ~~value~~ name → o/p → (10, 10)

call by reference :-



* o/p → (10, 5)

call by ~~value~~ name :-



o/p → (i, A[i]) → (10, 10)

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Ans:- The O/p will ~~not come~~ same always in call by value and call by reference.

Q:- procedure (a, b, c)

```
{
a=2;
c=a+b;
}
main();
{
x=1
y=5;
z=100;
P(x, x*y, z)
write(x, z);
}
```

x → x, x*y → x*y, c → z

x = 2.

z = 10

O/p → (2, 10).

x	x*y	z	y
~~2~~ 1	~~5~~ 10	~~100~~ ~~2~~ 10	5

First do diagram for actual arguments. Then the corresponding formal argument. If any changes occur in actual argument then check relative expression to change the value.

Ex:- when 'x' value changes in actual argument then check for relative expression x*y to change the value of 'x*y'. (relative to 'x')

Kiwal

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
main( )
{
int K=50, m=25;
  swap(K,m);
  print(K,m);
}
```

→ 50,5

K: 50 m: ~~25~~ → b=t, t=m

Q.

trace

```
Begin
  integer x;
  x=1;
  procedure p(y)
    integer x;
    x=2;
    print(y);
  end P
  p(x);
end.
```

x: 1

x_L: 2

x=1

actual prog

call by name → o/p → 1

call by text → o/p → 2

Call b text 12.0

```
integer i,j;
void p(integer x)
{
print(x+10);
  i=200;
  j=50;
print(x);
}
```

i: 200 (100), j: 50 (10), i+j: 110

(i+j) → 120

250

```
main( )
{
i=100
j=10
p(i+j);
}
```

call by name

no loi void so give default argu 110

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

50

call by text :- O/P → 120 and 250
Call by name = 120 and 250 .

25

Q:

```
integer i, j;
void P( integer x )
{
print (x+10);
int i = 200;
int j = 50;
print(x);
}
```

i+j

C.N 120

(C.T) 120

local variable.

C.N 110

C.T 250

```
main( )
{
i = 100;
j = 10;
p(i+j);
}
```

call by name (C.N) → 120, 110

(C.T) call by text → 120, 250

i	j
200	50

```
integer i, j;
void p(integer x)
{
print (x+10);
int i = 200;
int j = 50;
print(x);
}
```

120

global variable.

110

```
main( )
{
int i=100;
int j= 10;
P(i+j);
}
```

i	j
100	10

Call b text 120

no loi varia so give defa argu 110

actual arguments always main( ) function variable.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Call by name — 120, 110

call by text — 120, 110

* in call by text even if there is global variable it will give priority to ~~actual arg~~ local variable than global and actual arg. If there is no local variable give priority to actual arg. than global variable.

* <u>CALL BY NEED</u> (<u>Lazy evaluation</u>)

(i) Call by Need mechanism is a memorized version of <u>call by name</u>.

(ii) If the function argument is evaluated then that value is stored for subsequent uses.

```
integer i, j;                 i: 100 200      j: 10 50
void p (integer x)                                void main( )
{                                                 {
                        i+j                       i = 100
print (x+10);  ---->  120                         j = 10;
  i = 200                                         p(i+j);
  j = 50                  i+j=110                 }
print (x);
}
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

STATIC SCOPE AND DYNAMIC SCOPE

Referencing Environment :-

Referencing environment of statement is the collection of all variable which are visible in the statement.

Static scope :-

Referencing environment of a statement in a staticall scoped language. is the collection of all variables which are locally declared in its scope + All its ancestor variables which are visible in the statement.

$\boxed{\text{locally Declared + ancestor variables visible.}}$

Q- Find Referencing environment of the statement (1), 2, 3, 4, 5 in the below program using static scoping.

```
procedure Example
    A, B : integer;
    pro
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)


```
procedure Example
    A,B : integer;
    procedure Sub1
        x,y : integer;
        begin
                                    ---- (1)
    end
    procedure Sub2
        x : integer;
                                    ---- (2)
    procedure Sub 3
        x : integer
        begin
                                    ---- (3)
    end of sub 3
        begin
                                    ---- (4)
    end of sub 2
        begin (of example)
                                    ---- (5)
end (of example)
```

visibility :- If the same name variable of ancestor is not declared in a function then the variable is visible in b

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Statement	Referencing Environment
(1)	X, Y of Sub 1 + A, B of example. (A next to "")
(3)	X of sub 2 + ~~X of~~ A, B of example.
(3)	X of sub 3 + A, B of example.
(4)	X of sub 2 + A, B of example.
(5)	A, B of example.

Dynamic Scope :-

NOTE :- A subprogram is said to be active if its execution has started but not yet terminated.

→ Referencing environment of the statement in a dynamically scoped language is - the collection of

all variables which are declared locally to its scope +

All other variables of all other active programs which are visible in the statement.

Locally Declared + Active program variables which are visible.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Q:- Find Referencing environment of the statement 1, 2, 3 for the below program where the calling sequence is main → sub2 → sub1.

```
void sub1( )
{
int a, b;
        ——— (1)
}
void sub2( )
{
int b, c;
        ——— (2)
sub1( );
}
void main( )
{
int c, d;
        ——— (3)
sub2( );
}
```

main → sub2 → sub2

(Locally declared variable) → c, d b, c a, b

Stmt	Referencing environment
(1)	a, b of sub1 + c of sub2 + d of main
(2)	b, c of sub2 + d of main.
(3)	c, d of main.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

1, 2, 3 ance is

Q:- 1

Find the Referencing environment of the stmt 1, 2, 3 to the above program if the calling sequence is main → sub1 → sub2

28

Stmt	Referencing environment.
1	a, b of sub1 + c, d of main.
2	b, c of sub2 + a of sub1 + d of main.
3	c, d of main.

1st

main → sub1 → sub2.
(c, d) (a, b) (b, c)

→ sub2
↑
a, b

PROBLEM

Q:-

page-
Pre

main

Q:-

page 334
Pre 38

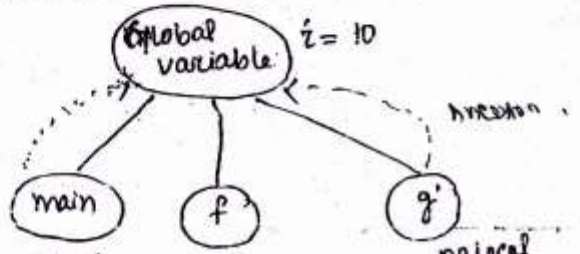
```
int i;
program main ()
{
 i = 10;
 call F( );
}
procedure F( )
{
 int i = 20;
 call g( );
}
procedure g( )
{
 print i;
```

Global variable i → main → f( ) → g( )
(i)

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

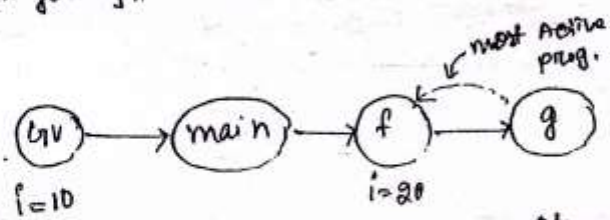
what is o/p if the variables are (a) statically scoped
(b) dynamically scoped.

Static scoping



So i value assign globally.

o/p → Ancestor's variable = 10



o/p = 20
most recently active program.

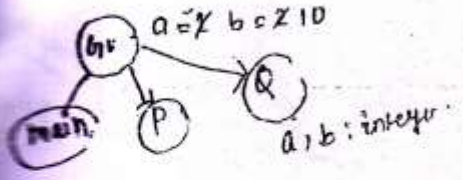
Q:-

```
a,b : integer;
procedure p( )
 {
 a = 5;
 b = 10;
 }
procedure q( )
 {
 a,b : integer;
 p( );
```

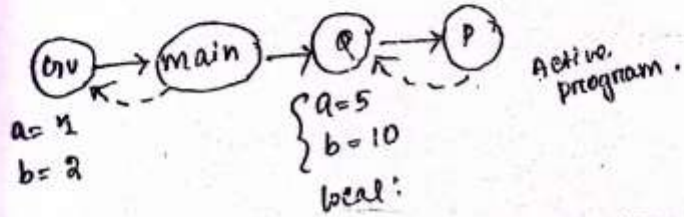
```
main ( )
 {
 a = 1;
 b = 2;
 q( );
 write(a,b);
 }
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

29



static scope: a=5, b=10



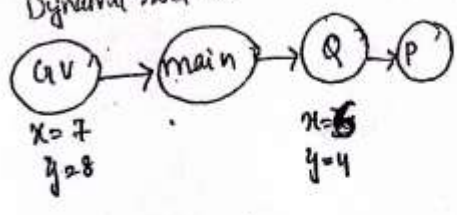
dynamic scope o/p → a=1, b=2.

* control at main see the other active program, only one is active program G.V. so take the value a=1, b=2.

```
x,y : integer
p(n : integer)
  {
  x = (n+2)/(n-3) ;
  }
Q( )
{
x, y : integer ;
  x = 3 ;
  y = 4 ;
  p(y)
  write(x) ;
}
```

```
main( )
{
x=7
y=8
Q
write(x) ;
}
```

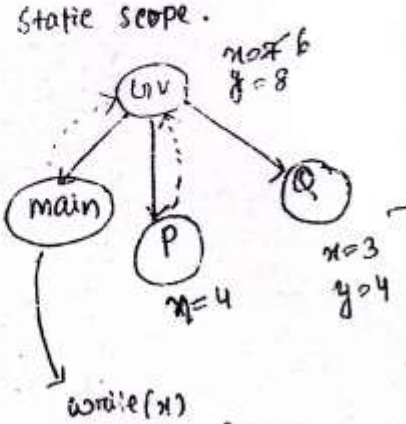
Dynamic scop.



O/p → 6, 7

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Static scope.



O/p → 3, 6.

write(x) = 3.

page-?
Q:

page-314
Q:3

```
n: integer
procedure w( x: integer)
{
x = x+1;
print (x);
}
procedure D
{
  n: integer;
  n = 3;
  w( n);
}
main ( )
{
n=10;
D;
}
```

n: 10 (global variable) 200

x: 100

100

n: 3 100 (local)

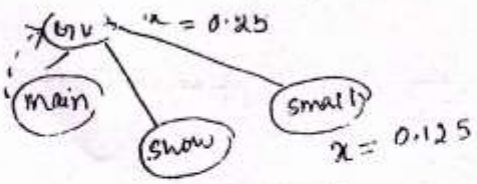
what is the O/p of above program if the parameter passing technique is call by reference and the variables are dynamically scoped.

Glv → main → D → w

n=10 n=3

∴ O/p → 4

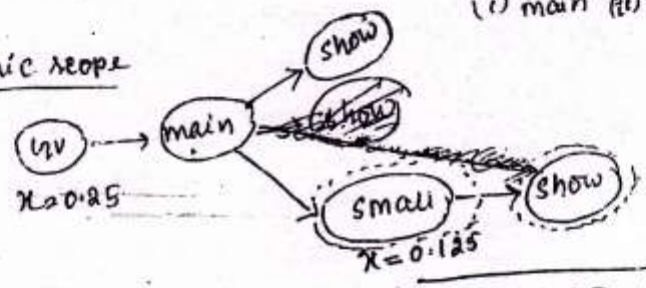
CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)59



$\boxed{\text{O/p} \rightarrow 0.25 \ , \ 0.25}$

here we called show() two times in in (i) main (ii) small.

dynamic scope



$\boxed{\text{O/p} \rightarrow 0.25 \ , 0.125}$

```
x : real ;
procedure show( )
 {
 print (x);
 }
procedure small( )
 {
 x : real ;
   x = 0.125 ;
   show( ) ;
 }
main
 {
 x=0.25
  show( ) ;
  small( ) ;
   }
```

O/P → 0.25 ← O/p → 0.25.

O/p → 0.25, 0.25.

Non static scope.

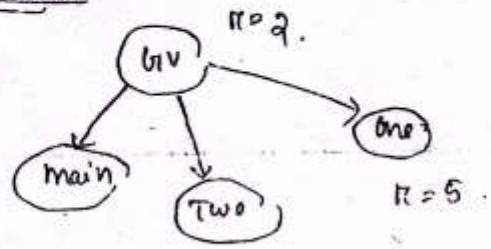
Dynamic scope

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)60

318
20

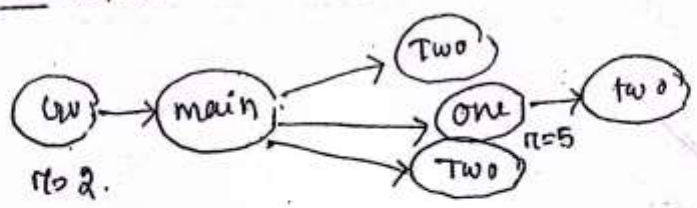
```
r: integer ;
procedure two( )
{
write (r);
}
Procedure one( )
{
r: integer ;
r=5;
two( );
}

main ( )
{
r=2 ;
two( );
one( );
two( );
}
```

Static scope



O/p → 2 , 2 ; 2

Dynamic scope

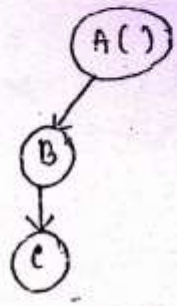


O/p → 2 , 5 , 2 .

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)61

Q: CREATE STATIC link pointers to the below program 31 where the calling sequence is A → B → C → B → C.

```
function A( )
{ int i;
  function B( )
  {
  int j;
    function C( )
    {
    int K;
      K = i+j;
      B( );
    }
    C( );
  }
  B( );
}
```



Rule:-

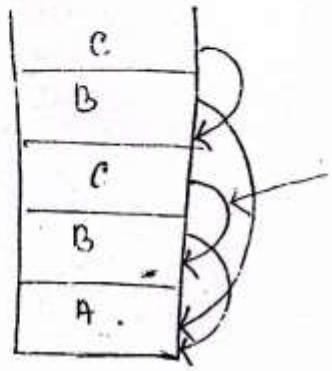
(i) each ~~iteration~~ activation record on the stack has a static link pointer to the Activation Record of most recent call to the routein which lexically ~~content~~ contains it.

→ In the above program 'B' is defined in 'A' so lexically 'A( )' contains 'B', 'C' Define with in B( ) so B( ) lexically contains 'C( )'.

'B( )' lexically contains C( ) OR C( ) Defined with in B( ).

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

$A \rightarrow B \rightarrow C \rightarrow B \rightarrow C$



C() is defined with in A() and B() call is but most recently B()

(STACK LINK POINTER.)

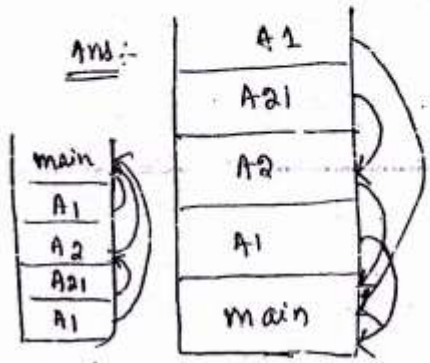
1 Binding occurs

LATE-2012

```
Program main
    variable
    procedure A1
        var. -
        Call A2;
    end A1
    procedure A2
        var - - -
        procedure A21;
            var - - ..
            Call A1;
        end A21
    end A2
    Call A1;
end main
```

2

Ans:-



3

Calling sequence is

main → A1 → A2 → A21 → A1

Create Static link pa

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)63

BINDING

Free occu...

$\sum_{i=0}^{m} \sum_{j=0}^{n} a_{ik} \, b_{kj}$

Binding occurance

$i : 0 \to m$
$j : 0 \to n$ } Binding variable

applied occurance.

$m :$
$n :$
$k :$ } Free variable.

Free occurance.

applied occurance.

Binding occurance.

$\int_0^{x} \int_0^{y} f(x,y) \, dx \, dy$

$dx : 0 \to x$
$dy : 0 \to y$ } Binding variable.

$* \; f(x,y)\Big]_0^2$ 0 to 2 value is applied in $f(x,y)$. so function f, x, y are applied variable.

Binding occurance

applied

$\forall_x \exists_y \; P(x,y)$

x : For all

y : There is exist some

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

Defination of Binding :-

→ Binding is an association between two things that is name, and the thing which it names.

→ In example-1 i and j are names which are associat with values 0 to m and 0 to n respectively. So these variables are called Binding variables.

→ A name which is not associating with any value is called free variable.

→ If any variable use the meaning of Binding variabl Then it is called Applied variable.

Binding Time :-

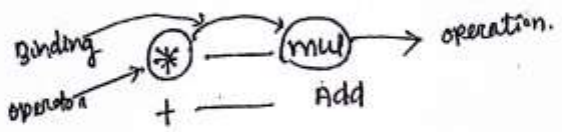
In programming language the Binding is created in 7 stages.

Static Binding:
- (i) language design time.
- (ii) Language implementation time
- (iii) program writting time.
- (iv) compile time.
- (v) link time.
- (vi) load time.

Dynamic Binding ← (vii) Run time.

(i) Language Design time.

Bind operator symbols (+, *, - and so on) to their operations. (add, mult, substraction...)



names of Premitive types declared inthe language design time.

(int , char , Float ...)

Language implementation time:-

Bind datatype (int, char, Float ...) to their possible range of values.

int -
char - 0 to 255

Binding.

Program writing time:-

programmer choose names of the variable, datastructures, algorithms at the time of writting the program.

Static int (ri) = (5) → name

ri = variable name
5 = value

char S [23]. → S = array name.

Compilation time:-

Bind the variables to their corresponding Datatypes.

Link time:-

Layout of whole program in the memory is finalized at link time. ~~the~~ i.e. names of separate modules to be linked at link time.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

declain · prototype.

Load time :-

Choice of Physical addresses are finalized at load time.

(static) int n = 5 .

Static variables are bound to the memory cell at load time.

Runtime :-

(i) All variables bound to their values at runtime

int n = 5

(ii) Size of strings are finalized.

char s[] = "hai"

(iii) program startup time, block entry time, procedure entry time, module entry time and statement execution time and finalized at runtime.

```
void f( )
{
  printf ("Hello");
}
```

Types of Binding

(i) Static Binding
(ii) Dynamic Binding.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)67

(i) Static Binding :-

A Binding is static if it [illegible] and remain unchange through out [illegible] called early Binding.

(ii) Dynamic Binding :-

A Binding is dynamic if it 1st occur during [illegible] or can change during the execution of the program. It is called late Binding.

* Storage allocation Techniques :-

(i) Static allocation
(ii) Stack allocation
(iii) Heap allocation. → Dynamic allocation.
malloc (size of (structnode));

(i) Static allocation :-

Function code, global variables, static variables, explicit constant, scalars, will be store into the static memory.

(ii) Stack allocation :-

All statements followed by function calls, ~~Subroutine~~ recurrsive call will be stored into the stack in the form of activation Record.

(iii) Heap allocation :-

→ All dynamic datastructure will be stored into the heap.

→ This memory is allocated and deallocated by explicite directives. (i.e. malloc, new, Delete ... etc)

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

POINTER TO STRUCTURE :-

```
struct tagname
  {
   member 1;
   member 2;
   :
   member n;
  }
```

Ex:- struct student → 12 bytes = (10+2) bytes.

```
  {
   int idno;        → size = 2bytes
   char name[10];   → 10 bytes
  };
```

→ tagname gives layout idea about data members of the structure.

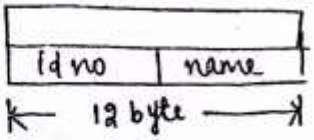
→ The keyword struct indicates that tagname is userdefined datatype.

* syntax to declare structure variable :-

struct tagname var1, var2, var3 . . . . varn;

ex:- struct student st; ['st' is a variable of student datatype usercdefind type]

struct student e1, e2;



* ACCESSING MEMBER of 'st'

st.idno; (by using selection operator)

st.name;

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)69

```
st[i].name ; or p[i].name ;
st[i].id no ; or p[i].id no ;
```
} ACCESSING MEMBERS OF Ith STRUCTURE.

page 3/2

NESTING OF STRUCTURE :-

we can declare one structure variable as a member into another structure. It is called nesting of structures.

```
struct date
{
int ~~date~~ dd;
int mm;
int yy;
};
```

} struct block should be declare before it is nested in another struct block.

```
struct employee
{
int idno;
struct date dob;
char name[10];
} e1, e2 = {10, 26, 06, 1981, "Sai"};
```

memory allocation :-

10	26	06	1981	Sai
	dd	mm	yy	
idno	dob			name

2 bytes ← → 6 bytes ← → 10 bytes.

← → 18 bytes.

S2
400

page :
Q.1

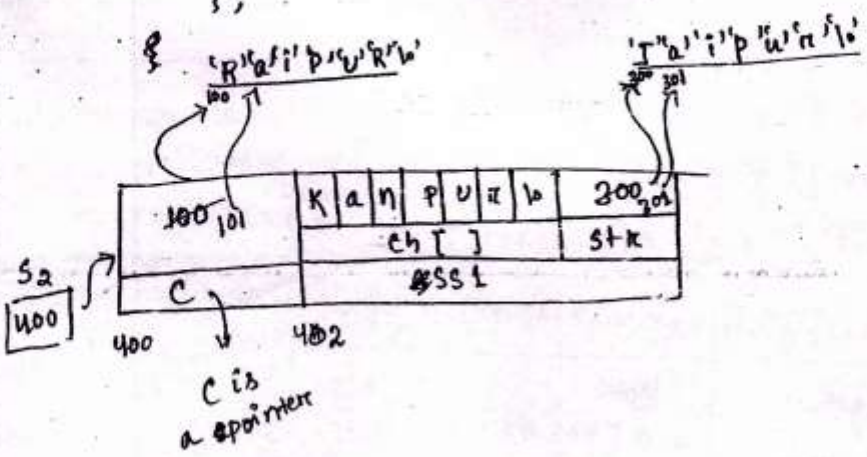
CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS) 70

page 342

(24)

```
main ()
{
struct a ——> 9
{
char ch[7]; ——> 2
char *str; ——> 2
};
struct b ——> 11
{
char *c; ——> 2
struct a ss1; ——> 9
};
```

```
struct b s2 = {"Raipur", "Kanpur",
               "Jaipur"}

printf ("%s %s", s2.c, s2.ss1.s1
                 ↓        ↓
                Raipur   Jaipur

printf ("%s %s", ++s2.c,
                 ↓
                aipur
                ++s2.ss1.str)
                 ↓
                aipur
}
```



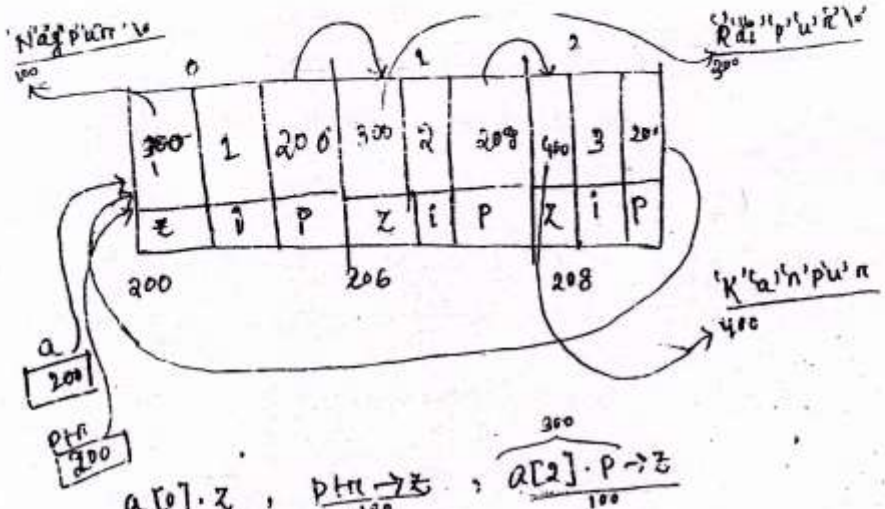
page 343

Q.-25

```
main ()
{
struct s1
{
char *z;
int i;
struct s1 *p;
};
```

```
struct s1 a[] = {{"Nagpur", 1, a+1},
                 {"Raipur", 2, a+2},
                 {"Kanpur", 3, a};

struct s1 *ptr = a;
printf ("%s %s %s", a[0].z,
                    100
                 ---> , a[2].p->z);
```

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)



a[0].z , ptr->z ; a[2].p->z

100 → Nagpur , 100 → Nagpur , 300 → 100 → Nagpur

page->343

Q:- 27

```
printf (" %s", ++(ptr->z));
printf (" %s", a[(++ptr)->i].z);
printf (" %s", a[--(ptr->p->i)].z);
```

ptr: 200 → 206

++(ptr->z)	a[(++ptr)->i].z	a[--(ptr->p->i)].z
agpur => ++(100) => 101 => agpur	~~kan~~ => a[206->i].z => a[2].z => Kanpur.	a[--(208->i)].z => a[--3].z => a[2].z => 400 => Kanpur.

* ptr->z changes to 101.

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS) 72

page 345

Q:- 30

'beebone'lo 100 | 'better'lo 200 | 'jungle'lo 300 | ancestor'lo 400 | brother'lo 500

0	1	2	3	4
5	100	4	200 / 201	6	300 / 301	8	400	7	500
i	c	i	c	i	c	i	c	i	c
600	604	608	612	616	620

P: 600 / 604 — 608

```
main
{
  struct test * P = St;
  P = P+1;
  ++P->c;                    -> 201
  printf("%s", (P++->c));    -> 604->c -> etter
608 <- printf("%c", * ++(P->c));  -> 'u' (608->c)
301 <- printf("%d", P[0].i);  -> 6
       printf("%s", P->c);   -> ungle
                     301
}
```

)]-z

]-z

if replace '%c' by '%s' then it will show error because we pass address by using '%s' but '*' is for data.

page 342

Q:- 23

ice'lo	Green'lo	cone'lo	please'lo
100	200	300	400

S: 500 — (500) — 504 — 506

St3	St2	St1	S
506	504	502	500

n+1 → 1

CREATED BY MISSION IMPOSSIBLE TO WANT GOOD QUALITY MATERIAL MAIL US: KAMAL.CHAUDHARY981@GMAIL.COM
Or OPEN THIS GROUP >>https://www.facebook.com/groups/1380084688879664 (STAY AWAY FROM FAKE NESS)

p
600
~~602~~
604

printf ("%s", **++p)

** 602

⇒ * 504

⇒ 300

done

printf ("%s", *--*++p+3)

p 602

--++p+3

(*--*++p)+3

(*--502)+3

(* 500)+3

100 + 3 = 103

no output printed.

printf ("%s", *p[-2]+3)

⇒ *p[-2]+3

⇒ *(*(p[-2]))+3

⇒ *(*(p-2x size of p+π))+3

⇒ *(*(604-2x2))+3

⇒ *(506)+3 ⇒ 400+3

⇒ 403

o/p → ase

pf("%s", p[-1][-1]+1)

p[-1][-1]+1

((p-1x size of p+π) - 1x size of data)

+ 1

⇒ *(*(p-2)-1x2)+2

⇒ *(*602-2)+2

⇒ *(504-2)+1

⇒ *502+1

⇒ ~~reen~~ 201

⇒ reen