Sum of $1^2+2^2+3^2 \cdots + 10^2 = \frac{n(n+1)(2n+1)}{6}$

$1+2+3+\cdots+n = \frac{n(n+1)}{2}$

Function

Function declaration is called function prototype.

Function call is called invocation.

If prototype is not defined and directly defination is given after main function then it is assomed by default function return int. ⟶ if type did not match with default → warning type-mismatch

Gate 2017

Consider the C prog. frg. below which is meant to divide x by y. using repeated subtraction. The variable x, y, q and r are all unsigned int.

```
while (r >= y){
    r = r - y;
    q = q + 1;
}
```

⟶ q = quotient, r = remender. y = divisor ∴ x = dividend

which of the following conditions on the variable x, y, q, r before execution of the fragment will ensure that the loop terminates. in a state satisfying the condition $x == (y*q+r)$

∴ $x = y*q+r$

$x = r,\ q = 0,\ y > 0$

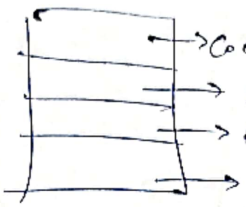
```
int a1(int z){
int b1(int x){
int main(){
    printf("%d%d", a1(2), b1(4));
}
```

```
printf("4"); return z;}
printf("8"); return x;}
```

→ output

8724

Memory Segment →



Activation Record :- The information regarding execution of a function is maintained in record format.

Local variable stored.

Storage class
- auto ✓
- static ✓
- Extern
- Register

The default storage class for Local variable is auto

Stack

If Local variable not initialized then it containing garbage value

Global variable automatically initialized to '0'.

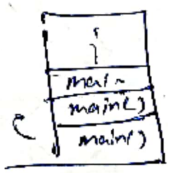
The default storage class for global variable is 'static'

Global variable is stored in static data in Memory segment

Recursion

```
int main(){
    main();
}
```

→

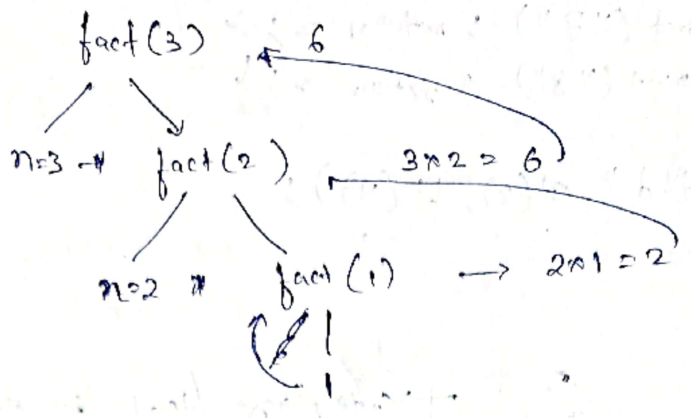


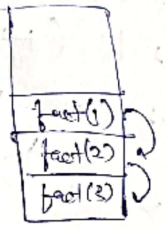
∴ Stack's allocated memory exhausted.
Stack overflow
Abnormal termination.

factorial

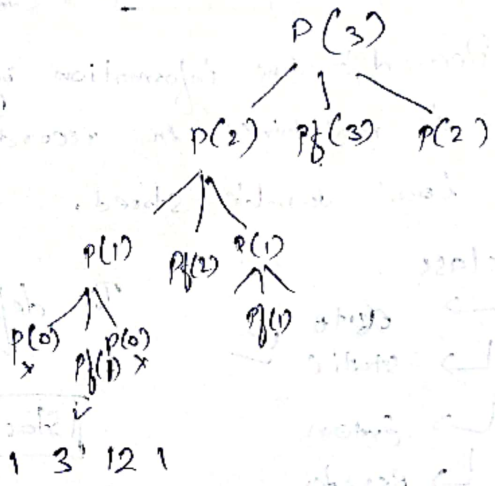
```
if fact(int n){
    if (n==0 || n==1)
        return 1;
    else
        return n * fact(n-1);
}
```

} Base condition.






```
void print (int n){
    if (n<=0) return;
    Print (n-1);
    printf ("%d", n);
    Print (n-1);
}
int main (){
    Print (3);
    return 0;
}
```



1 2 1 3 1 2 1

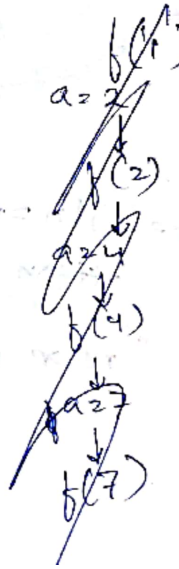
#
```
int foo ( int n){
    if (n<=9) return n;
    else
        return n%10 + foo(n/10);
}
int main(){
    Printf ("%d", foo (12345));
    return 0;
}
```

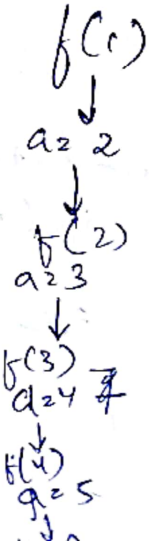
15

foo (12345)
5 + foo (1234) 10 = 15
4 + foo (123) 6 = 10
3 + foo (12) 3 = 6
2 + f(1) 1 = 3

#
```
int fun (int a){
    static int i;
    if (a >= 5) return a;
    a = a+1;
    i++
    return fun (a);
}
```

fun(1) 5.





1 2 1

Register Storage class

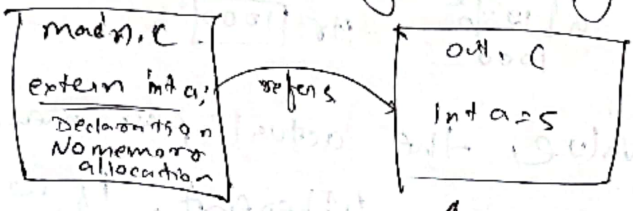
register int i = 100; ← compiler allocates register faster.

Extern Storage Class.

increase the scope of variable of variable, in file system.

'extern' Keyword

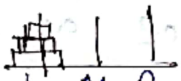
extern only on global variable



If not ~~declare~~ define then error on extern as it will not found the value.

to find in greater scopes.

Tower of Hanoi: $O(2^n)$



TOH (L, M, R, n)

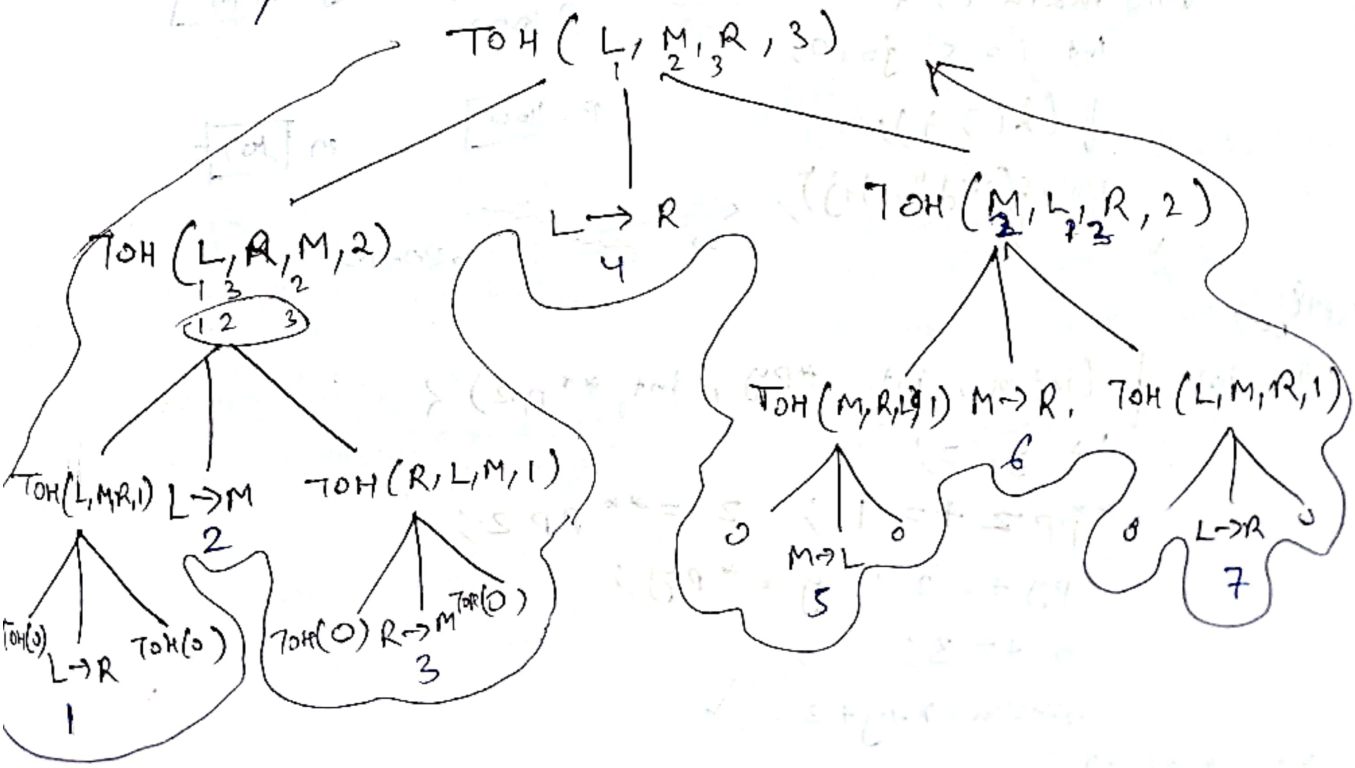
Move 3 disk from L to R using M. → if (n>0){

TOH (L, R, M, n-1)

L → R [The disk in L will get exposed and move it to R)

TOH (M, L, R, n-1)

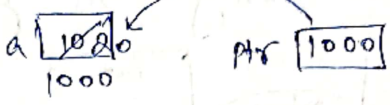
}



Pointer -> Call by reference.

```
Void foo (int *);
int main (){
    int a=10;
    foo(&a);
    printf("%d",a);
}
```

```
Void foo (int *ptr){
    *ptr = *ptr + 10;
}
```

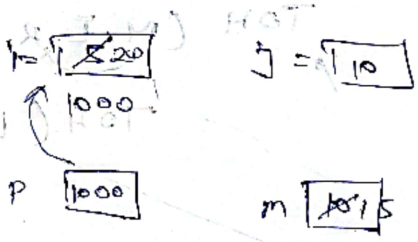


In Case of Call by value, the actual paramenter and the formal parameter is different, if we do any change in function, no change will be there.

In Call by reference as the pointer is linked with actual parameter, the value gets changed..

GATE

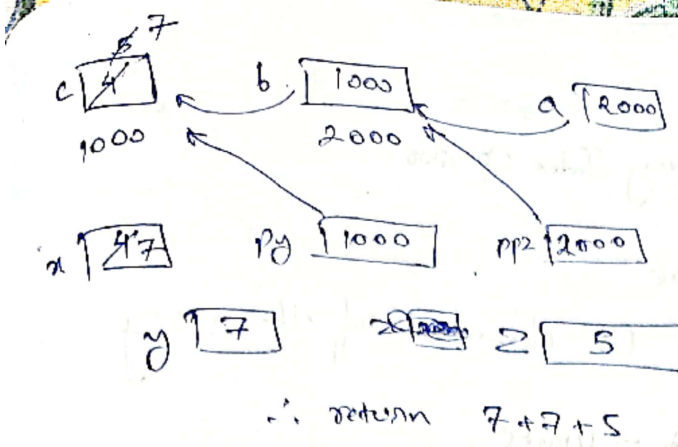
```
# Void f (int *p, int m){
    m = m+5;
    *p = *p + m;
    return;
}
Void main (){
    int i=5, j=10;
    f(&i, j);
    printf("%d", i+j);
}
```



30 ← 20+10

GATE 2008

```
# int f (int x, int *py, int **ppz){
    int y, z;
    **ppz += 1;  z = **ppz;
    *py += 2;  y = *py;
    x += 3;
    return x+y+z;
}
Void main(){
    int c, *b, **a;
    c = 4;  b = &c;  a = &b;
    printf("%d", f(c, b, a));
}
```



∴ return 7+7+5 = 19

Function returns Pointer.

Prototype: int foo(int); foo is a function which takes 1 integer Array argument & returns an integer.

Prototype: int *foo(int); foo is a function which takes 1 integer argument & returns an integer Pointer

Array of pointer. int (*p)[]; Pointer-Array.

Problem

```
int *bar(int);
int main(){
    int a=20;
    int *ptr;
    ptr = bar(a);
    printf("%d", *ptr);
}
```

```
int *bar(int b){
    int i = 20;
    b = i+20; → 40 but no use
    return &i;
}
```

a 20

b 20 (40)

i 20 (1000)

ptr 1000

But after the function terminates this 'i' local variable gets destroyed ∴ ptr will have the address which is deallocated/destroyed ∴ Warning and No print as compiler will not allow

Dangling Pointer problem →

Pointer holding address of a variable which don't exist.

Segmentation fault (run time error)

```
int *ptr;
*ptr = 5;
```

s[] → %s.

```
char s = "sss";
char s = 'sss';
  ↳ s = 's'; %c
```