

class 1

Recursion

Topic

- ① What is Recursion
- ② Example of Recursion
- ③ Tracing of Recursion
- ④ Stack used in Recursion
- ⑤ Time complexity
- ⑥ Recurrence Relation

function:

```
void fun(int n){  
    1---  
    2---  
    3---}
```

```
int main(){  
    1---  
    2---  
    3--- fun(n);  
    4---  
    5---}

```
3
4
5
```


```

Recursion

```
type fun(para)  
{  
    if (base case)  
    {  
    }
```

```
(a) int z = fun(para);  
    3---  
    {  
    }  
    (b) int z;  
        3---  
        {  
        }  
        (c) int z;
```

Tracing
Stack
Recurrence Relation

```
void fun(int n)
```

```
{
```

```
    if(n > 0)
```

```
{
```

```
    print(n);
```

```
    fun(n-1);
```

```
}
```

```
{
```

```
fun(3)
```

```
3
```

```
fun(2)
```

```
2
```

```
fun(1)
```

```
1
```

```
fun(0)
```

```
0
```

```
int main()
```

```
{
```

```
    int n=3;
```

```
    fun(n);
```

```
}
```

```
{
```

```
O/P: 3, 2, 1
```

```
void fun(int n)
```

```
{
```

```
    if(n > 0):
```

```
{
```

```
    fun(n-1);
```

```
    print(n);
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

```
{
```

Class 2

Recursion

void fun(int n)

$$\left\{ \begin{array}{l} f(n) \\ g(n) \end{array} \right\}_{n \geq 0}$$

1. calling time ← Ascending
 2. $\text{fun}(n-1) * 2;$
 3. returning time ↑ ← Descending

3

Class-4 : Every statement take 1 unit of time.

void fun(int n)

۳

if ($n > 0$)

{ print(n); → 1

fun(n-1);

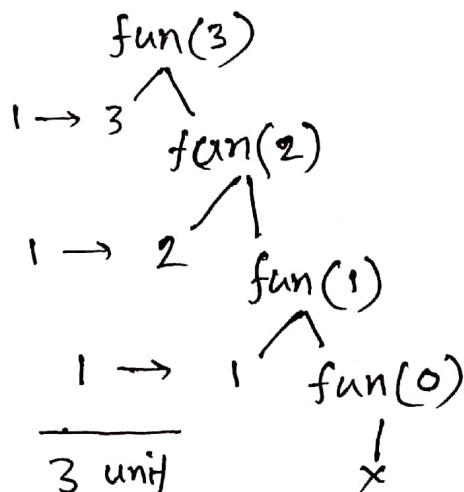
3

```
int main()
```

1

int n=3;

3



Value of n $O(n)$
 n unit

[put book in table n'
book's take n' unit of time

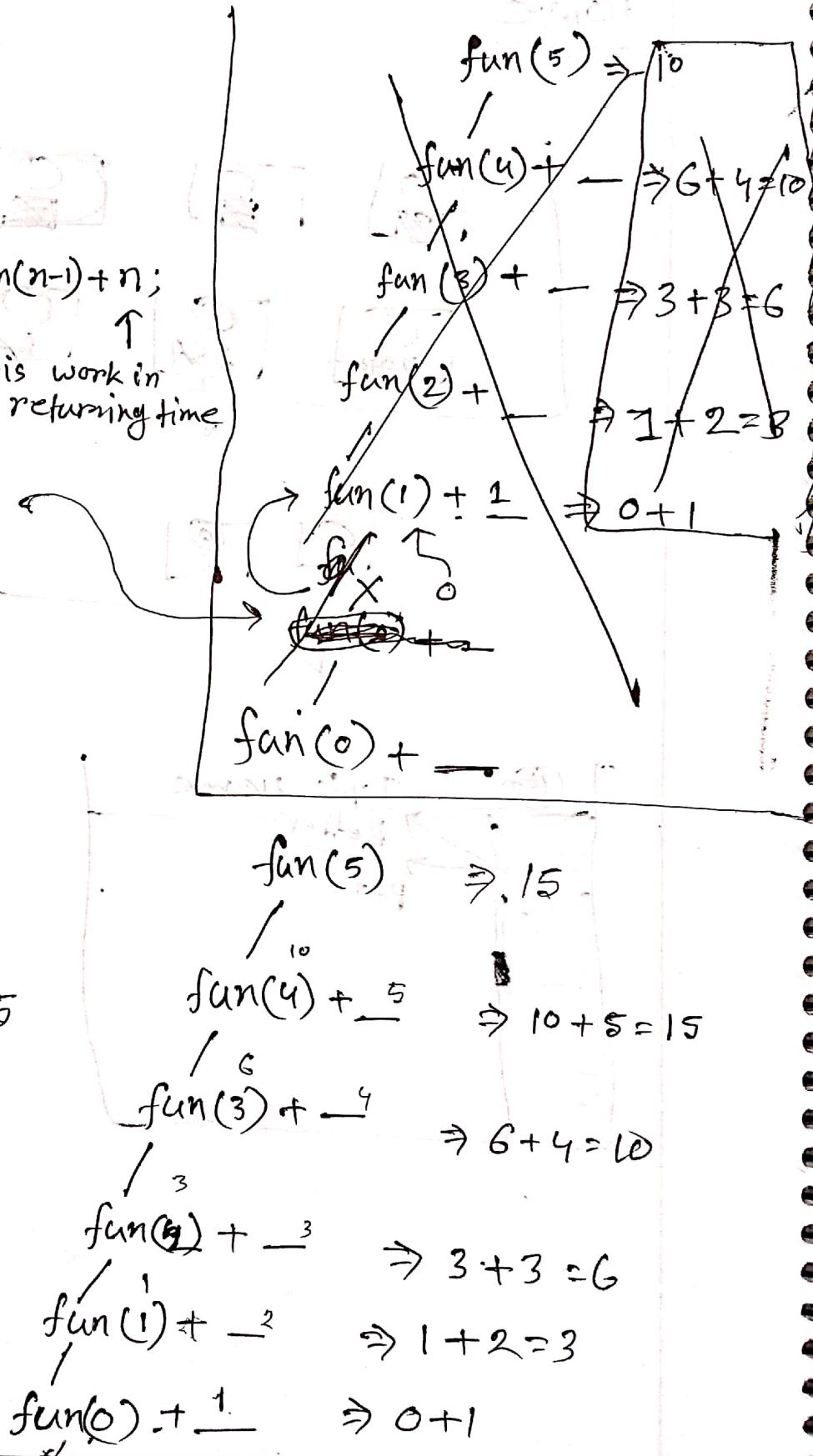
static Variable in Recursion

class - 8 :

```
int fun(int n)
{
    if (n > 0)
    {
        return fun(n-1) + n;
    }
    else
        return 0;
}
```

```
int main()
{
    int a = 5;
    fun(a);
}
```

$n = 5$
 $n = 4$
 $n = 3$
 $n = 2$
 $n = 1$



static \Rightarrow like global variable

S - static

int fun(int n)

{ static int n=0;

if(n>0)

{ n++;

return fun(n-1)+n;

}

return 0;

int main()

{

int a=5;

fun(a);

}

(a) static

(a) failing

i--;

}

(a) user

(a) O-> S-0

(a) T

fun(5)

n=0

fun(4)

n=1

fun(3)

n=2

fun(2)

n=3

fun(1)

n=4

fun(0)

n=5

fun(-1)

n=6

fun(-2)

fun(-3)

fun(-4)

fun(-5)

fun(-6)

fun(-7)

fun(-8)

fun(-9)

fun(-10)

fun(-11)

fun(-12)

fun(-13)

fun(-14)

fun(-15)

fun(-16)

fun(-17)

fun(-18)

fun(-19)

fun(-20)

fun(-21)

fun(-22)

fun(-23)

fun(-24)

fun(-25)

fun(-26)

fun(-27)

fun(-28)

fun(-29)

fun(-30)

fun(-31)

fun(-32)

fun(-33)

fun(-34)

fun(-35)

fun(-36)

fun(-37)

fun(-38)

fun(-39)

fun(-40)

fun(-41)

fun(-42)

fun(-43)

fun(-44)

fun(-45)

fun(-46)

fun(-47)

fun(-48)

fun(-49)

fun(-50)

fun(-51)

fun(-52)

fun(-53)

fun(-54)

fun(-55)

fun(-56)

fun(-57)

fun(-58)

fun(-59)

fun(-60)

fun(-61)

fun(-62)

fun(-63)

fun(-64)

fun(-65)

fun(-66)

fun(-67)

fun(-68)

fun(-69)

fun(-70)

fun(-71)

fun(-72)

fun(-73)

fun(-74)

fun(-75)

fun(-76)

fun(-77)

fun(-78)

fun(-79)

fun(-80)

fun(-81)

fun(-82)

fun(-83)

fun(-84)

fun(-85)

fun(-86)

fun(-87)

fun(-88)

fun(-89)

fun(-90)

fun(-91)

fun(-92)

fun(-93)

fun(-94)

fun(-95)

fun(-96)

fun(-97)

fun(-98)

fun(-99)

fun(-100)

fun(-101)

fun(-102)

fun(-103)

fun(-104)

fun(-105)

fun(-106)

fun(-107)

fun(-108)

fun(-109)

fun(-110)

fun(-111)

fun(-112)

fun(-113)

fun(-114)

fun(-115)

fun(-116)

fun(-117)

fun(-118)

fun(-119)

fun(-120)

fun(-121)

fun(-122)

fun(-123)

fun(-124)

fun(-125)

fun(-126)

fun(-127)

fun(-128)

fun(-129)

fun(-130)

fun(-131)

fun(-132)

fun(-133)

fun(-134)

fun(-135)

fun(-136)

fun(-137)

fun(-138)

fun(-139)

fun(-140)

fun(-141)

fun(-142)

fun(-143)

fun(-144)

fun(-145)

fun(-146)

fun(-147)

fun(-148)

fun(-149)

fun(-150)

fun(-151)

fun(-152)

fun(-153)

fun(-154)

fun(-155)

fun(-156)

fun(-157)

fun(-158)

fun(-159)

fun(-160)

fun(-161)

fun(-162)

fun(-163)

fun(-164)

fun(-165)

fun(-166)

fun(-167)

fun(-168)

fun(-169)

fun(-170)

fun(-171)

fun(-172)

fun(-173)

fun(-174)

fun(-175)

fun(-176)

fun(-177)

fun(-178)

fun(-179)

fun(-180)

fun(-181)

fun(-182)

fun(-183)

fun(-184)

fun(-185)

fun(-186)

fun(-187)

fun(-188)

fun(-189)

fun(-190)

fun(-191)

fun(-192)

fun(-193)

fun(-194)

fun(-195)

fun(-196)

fun(-197)

fun(-198)

fun(-199)

fun(-200)

fun(-201)

fun(-202)

fun(-203)

fun(-204)

fun(-205)

fun(-206)

fun(-207)

fun(-208)

fun(-209)

fun(-210)

fun(-211)

fun(-212)

fun(-213)

fun(-214)

fun(-215)

fun(-216)

fun(-217)

fun(-218)

fun(-219)

fun(-220)

fun(-221)

fun(-222)

fun(-223)

fun(-224)

fun(-225)

fun(-226)

fun(-227)

fun(-228)

fun(-229)

fun(-230)

fun(-231)

fun(-232)

fun(-233)

fun(-234)

fun(-235)

fun(-236)

fun(-237)

fun(-238)

fun(-239)

fun(-240)

fun(-241)

fun(-242)

fun(-243)

fun(-244)

fun(-245)

fun(-246)

fun(-247)

fun(-248)

fun(-249)

fun(-250)

fun(-251)

fun

Class - 8

Types of Recursion

1. Tail Recursion
2. Head Recursion
3. Tree Recursion
4. Indirect Recursion
5. Nested Recursion

1. Tail Recursion:

The Recursive function if the last statement is the Recursive call then it is called Tail Recursion.

```
void fun(int n)
{
    if (n > 0)
    {
        print(n);
        fun(n-1);
    }
}
```

fun(3)

convert to loop

```
void fun(int n)
{
    while (n > 0)
    {
        print(n);
        n--;
    }
}
```

fun(3)

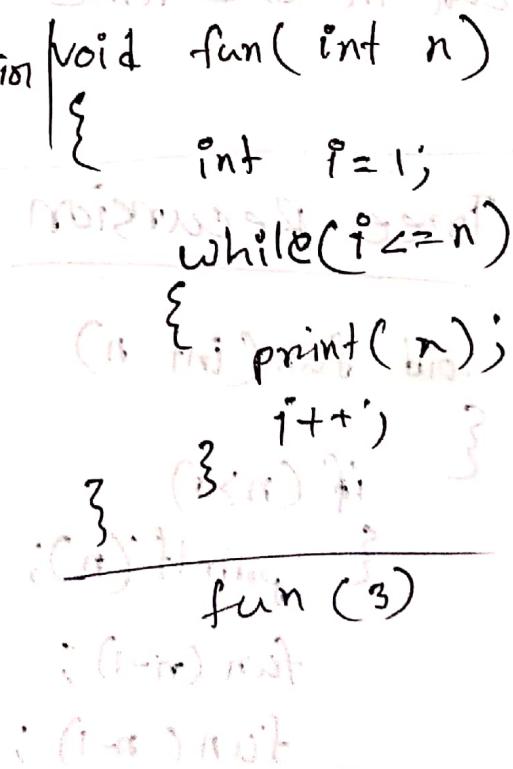
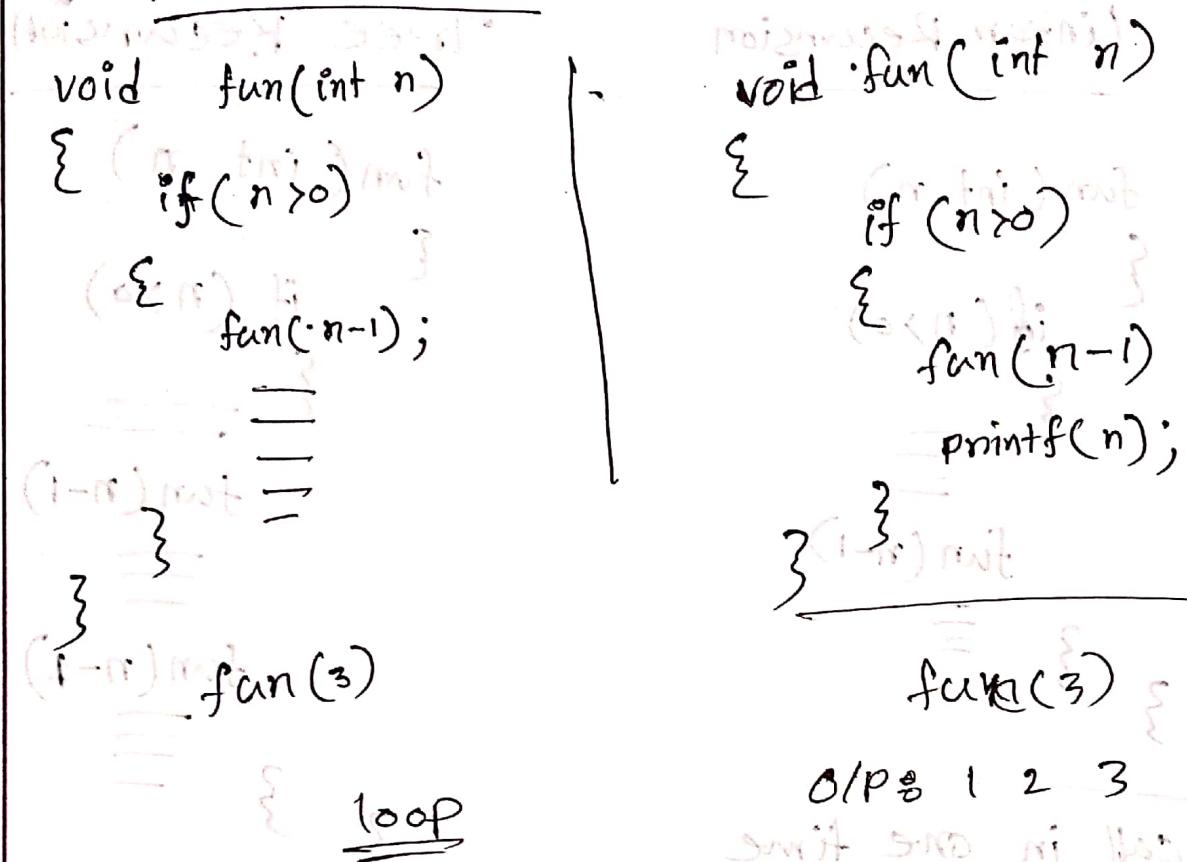
Time $\rightarrow O(n)$

Space $O(n)$

$O(1)$

class-9

2. Head Recursion



Tree Recursion

Linear Recursion

```
fun(int n)
{
    if(n>0)
        fun(n-1)
```

call in one time

Tree Recursion

```
fun(int n)
{
    if(n>0)
        fun(n-1)
        fun(n-1)
```

call in two time

Tree Recursion

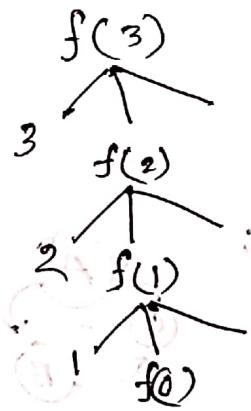
```
void fun(int n)
{
    if(n>0)
        printf("%d");
        fun(n-1);
        fun(n-1);
```

Tree Recursion

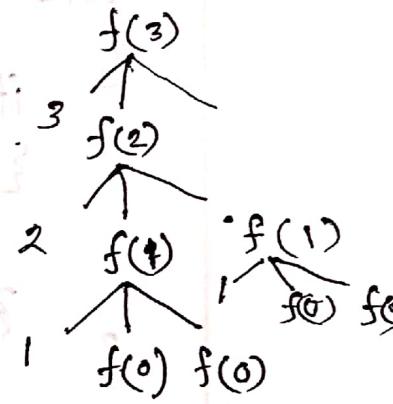
tracing tree

```
void fun(int n)
{
    if(n > 0)
        print(n);
    fun(n-1);
    fun(n-1);
}
```

First call



First with second call



fun(3)

(e1) start

(e2) first

(e3) print

(e4) fun(2)

(e5) fun(1)

(e6) fun(0)

(e7) fun(0)

(e8) fun(1)

(e9) fun(2)

(e10) fun(3)

(e11) fun(4)

(e12) fun(5)

(e13) fun(6)

(e14) fun(7)

(e15) fun(8)

(e16) fun(9)

(e17) fun(10)

(e18) fun(11)

(e19) fun(12)

(e20) fun(13)

(e21) fun(14)

(e22) fun(15)

(e23) fun(16)

(e24) fun(17)

(e25) fun(18)

(e26) fun(19)

(e27) fun(20)

(e28) fun(21)

(e29) fun(22)

(e30) fun(23)

(e31) fun(24)

(e32) fun(25)

(e33) fun(26)

(e34) fun(27)

(e35) fun(28)

(e36) fun(29)

(e37) fun(30)

(e38) fun(31)

(e39) fun(32)

(e40) fun(33)

(e41) fun(34)

(e42) fun(35)

(e43) fun(36)

(e44) fun(37)

(e45) fun(38)

(e46) fun(39)

(e47) fun(40)

(e48) fun(41)

(e49) fun(42)

(e50) fun(43)

(e51) fun(44)

(e52) fun(45)

(e53) fun(46)

(e54) fun(47)

(e55) fun(48)

(e56) fun(49)

(e57) fun(50)

(e58) fun(51)

(e59) fun(52)

(e60) fun(53)

(e61) fun(54)

(e62) fun(55)

(e63) fun(56)

(e64) fun(57)

(e65) fun(58)

(e66) fun(59)

(e67) fun(60)

(e68) fun(61)

(e69) fun(62)

(e70) fun(63)

(e71) fun(64)

(e72) fun(65)

(e73) fun(66)

(e74) fun(67)

(e75) fun(68)

(e76) fun(69)

(e77) fun(70)

(e78) fun(71)

(e79) fun(72)

(e80) fun(73)

(e81) fun(74)

(e82) fun(75)

(e83) fun(76)

(e84) fun(77)

(e85) fun(78)

(e86) fun(79)

(e87) fun(80)

(e88) fun(81)

(e89) fun(82)

(e90) fun(83)

(e91) fun(84)

(e92) fun(85)

(e93) fun(86)

(e94) fun(87)

(e95) fun(88)

(e96) fun(89)

(e97) fun(90)

(e98) fun(91)

(e99) fun(92)

(e100) fun(93)

(e101) fun(94)

(e102) fun(95)

(e103) fun(96)

(e104) fun(97)

(e105) fun(98)

(e106) fun(99)

(e107) fun(100)

(e108) fun(101)

(e109) fun(102)

(e110) fun(103)

(e111) fun(104)

(e112) fun(105)

(e113) fun(106)

(e114) fun(107)

(e115) fun(108)

(e116) fun(109)

(e117) fun(110)

(e118) fun(111)

(e119) fun(112)

(e120) fun(113)

(e121) fun(114)

(e122) fun(115)

(e123) fun(116)

(e124) fun(117)

(e125) fun(118)

(e126) fun(119)

(e127) fun(120)

(e128) fun(121)

(e129) fun(122)

(e130) fun(123)

(e131) fun(124)

(e132) fun(125)

(e133) fun(126)

(e134) fun(127)

(e135) fun(128)

(e136) fun(129)

(e137) fun(130)

(e138) fun(131)

(e139) fun(132)

(e140) fun(133)

(e141) fun(134)

(e142) fun(135)

(e143) fun(136)

(e144) fun(137)

(e145) fun(138)

(e146) fun(139)

(e147) fun(140)

(e148) fun(141)

(e149) fun(142)

(e150) fun(143)

(e151) fun(144)

(e152) fun(145)

(e153) fun(146)

(e154) fun(147)

(e155) fun(148)

(e156) fun(149)

(e157) fun(150)

(e158) fun(151)

(e159) fun(152)

(e160) fun(153)

(e161) fun(154)

(e162) fun(155)

(e163) fun(156)

(e164) fun(157)

(e165) fun(158)

(e166) fun(159)

(e167) fun(160)

(e168) fun(161)

(e169) fun(162)

(e170) fun(163)

(e171) fun(164)

(e172) fun(165)

(e173) fun(166)

(e174) fun(167)

(e175) fun(168)

(e176) fun(169)

(e177) fun(170)

(e178) fun(171)

(e179) fun(172)

(e180) fun(173)

(e181) fun(174)

(e182) fun(175)

(e183) fun(176)

(e184) fun(177)

(e185) fun(178)

(e186) fun(179)

(e187) fun(180)

(e188) fun(181)

(e189) fun(182)

(e190) fun(183)

(e191) fun(184)

(e192) fun(185)

(e193) fun(186)

(e194) fun(187)

(e195) fun(188)

(e196) fun(189)

(e197) fun(190)

(e198) fun(191)

(e199) fun(192)

(e200) fun(193)

(e201) fun(194)

(e202) fun(195)

(e203) fun(196)

(e204) fun(197)

(e205) fun(198)

(e206) fun(199)

(e207) fun(200)

(e208) fun(201)

(e209) fun(202)

(e210) fun(203)

(e211) fun(204)

(e212) fun(205)

(e213) fun(206)

(e214) fun(207)

(e215) fun(208)

(e216) fun(209)

(e217) fun(210)

(e218) fun(211)

(e219) fun(212)

(e220) fun(213)

(e221) fun(214)

(e222) fun(215)

(e223) fun(216)

(e224) fun(217)

(e225) fun(218)

(e226) fun(219)

(e227) fun(220)

(e228) fun(221)

(e229) fun(222)

(e230) fun(223)

(e231) fun(224)

(e232) fun(225)

(e233) fun(226)

(e234) fun(227)

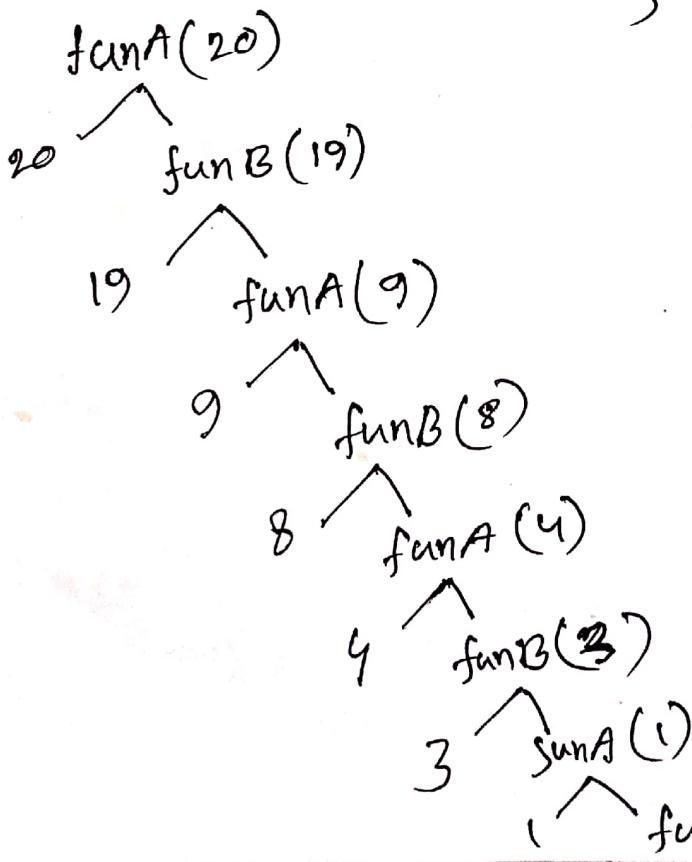
(e235) fun(2

Indirect type of Recursion

Circular fashion

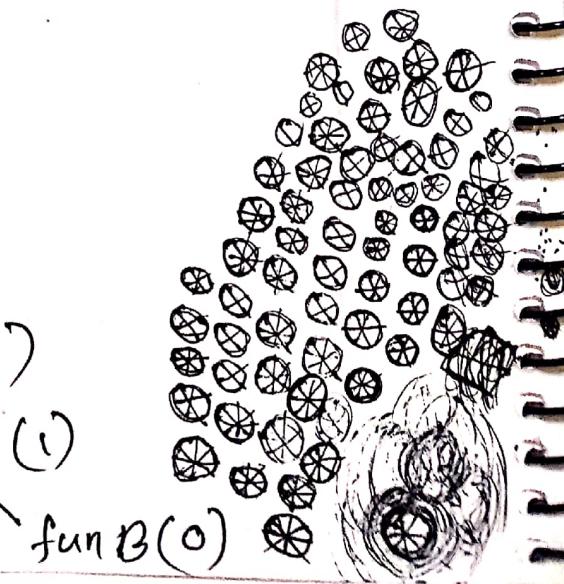
```
void A(int n)
{
    if(→)
    {
        ≡
        } B(n-1);
}
```

```
void B(int n)
{
    if(→)
    {
        ≡
        } A(n-3)
}
```



```
void funA(int n)
{
    if(n>0)
    {
        printf("%d",n);
        funB(n-1);
    }
}
```

```
void funB(int n)
{
    if(n>1)
    {
        printf("%d",n);
        funB(n/2);
    }
}
```



class-14

Nested Recursion

```
void fun(int n)
{
    if(n == 1)
        {
            = fun(fun(n-1));
        }
}
```

Recursion inside the recursion when inside recursion value obtain then outer recursion call made.

```
int fun(int n)
{
    if(n > 100)
        return n-10;
    else
        return fun(fun(n+1));
}
```

```
fun(95)
  fun(fun(95+1))
    fun(96)
      fun(fun(96+1))
        fun(97)
          fun(fun(97+1))
            fun(98)
              fun(fun(98+1))
                fun(99)
                  fun(fun(99+1))
                    fun(100)
```

10
11

Sum of First 'n' Natural Numbers

Class - 16

$$1 + 2 + 3 + 4 + 5 + 6 + \dots$$

$$1 + 2 + 3 + \dots + n$$

$$\text{sum}(n) = 1 + 2 + 3 + \dots + (n-1) + n$$

$$\text{sum}(n) = \text{sum}(n-1) + n$$

$$\text{sum}(n) = \begin{cases} 0 & \text{if } n=0 \\ \text{sum}(n-1) + n & \text{if } n>0 \end{cases}$$

```
int sum(int n)
{
    if (n==0)
    {
        return 0;
    }
    else
    {
        return sum(n-1) + n;
    }
}
```

$$\frac{n(n+1)}{2}$$

$O(n)$

space $O(n)$

$$\begin{aligned}
\text{sum}(5) &= 15 \\
\text{sum}(4) + 5 &= 10 + 5 = 15 \\
\text{sum}(3) + 4 &= 6 + 4 = 10 \\
\text{sum}(2) + 3 &= 3 + 3 = 6 \\
\text{sum}(1) + 2 &= 1 + 2 = 3 \\
\text{sum}(0) + 1 &= 0 + 1
\end{aligned}$$

class-18

Factorial of a Number

$$n! = 1 * 2 * 3 * \dots * n$$

$$5! = 1 * 2 * 3 * 4 * 5 = 120$$

$$0! = 1$$

$$1! = 1$$

$$\text{fact}(n) = 1 * 2 * 3 * \dots * (n-1) * n$$

$$\text{fact}(n) = \text{fact}(n-1) * n$$

$$\text{fact}(n) = \begin{cases} 1 & n=0 \\ \text{fact}(n-1) * n & n>0 \end{cases}$$

int fact(int n)

{ if(n==0)

 return 1;

else

 return fact(n-1) * n;

$$(n=5) \Rightarrow \text{fact}(5) \rightarrow 120$$

$$\text{fact}(4) \rightarrow 4 * 3 * 2 * 1$$

$$\text{fact}(3) \rightarrow 3 * 2 * 1$$

$$\text{fact}(2) \rightarrow 2 * 1$$

$$\text{fact}(1) \rightarrow 1$$

$$\text{fact}(0) \rightarrow 1$$

$O(n)$

space(n)

Σ $\Theta(n)$

Σ $\Theta(n)$

Σ $\Theta(n)$

7

Exponent $(m)^n$ power

class 20

$$2^5 = 2 * 2 * 2 * 2 * 2 = 32$$

$$m^n = m * m * m * \dots \text{for } n \text{ times}$$

$$\text{pow}(m, n) = \underbrace{m * m * m * \dots}_{(n-1) \text{ times}} * m$$

$$\text{pow}(m, n) = \text{pow}(m, n-1) * m$$

$$\text{pow}(m, n) = \begin{cases} 1 & n=0 \\ \text{pow}(m, n-1) * m & n>0 \end{cases}$$

int pow(int m, int n)

```
{ if (n == 0)
    return 1;
```

```
return pow(m, n-1) * m;
```

}

$$128 = (2^2)^4$$

$$= (2 * 2)^4$$

$$2^9 = 2 * (2^4)^4$$

power reduce
by half

$$\text{pow}(2, 9)$$

$$\text{pow}(2, 8) * 2$$

$$\text{pow}(2, 7) * 2$$

$$\text{pow}(2, 6) * 2$$

$$\text{pow}(2, 5) * 2$$

$$\text{pow}(2, 4) * 2$$

$$\text{pow}(2, 3) * 2$$

$$\text{pow}(2, 2) * 2$$

$$\text{pow}(2, 1) * 2$$

$$\text{pow}(2, 0) * 2$$

```

int pow (int m, int n)
{
    if (n == 0)
        return 1;
    if (n % 2 == 0)
        return pow(m * m, n / 2);
    else
        return m * pow(m * m, (n - 1) / 2);
}

```

$$pow(2, 9) \Rightarrow 2^9$$

$$\downarrow \\ 2 * pow(2^4, 4); \Rightarrow 2 * 2^8$$

$$\downarrow \\ pow(2^{16}, 2) \quad 2^{16} \Rightarrow 2^8$$

$$\downarrow \\ pow(2^8, 1) \Rightarrow 2^8$$

$$\downarrow \\ 2^8 * pow(2^{16}, 0) \Rightarrow 2^8 * 1$$

1

Taylor series e^n

Class 22

$$e^n = 1 + \frac{n}{1} + \frac{n^2}{2!} + \frac{n^3}{3!} + \frac{n^4}{4!} + \dots + n \text{ terms}$$

$$\text{sum}(n-1)*n \Rightarrow \text{sum}(n) = 1 + 2 + 3 + \dots + n$$

$$\text{fact}(n-1)*n \Rightarrow \text{fact}(n) = 1 * 2 * 3 * 4 * \dots * n$$

$$\text{pow}(n, n-1)*n \Rightarrow \text{pow}(n, n) = n * n * n * \dots * n \text{ times}$$

($e(n, 4)$)

$$e(n, 3) \Rightarrow 1 + \frac{n}{1} + \frac{n^2}{2!} + \frac{n^3}{3!} + \left(\frac{n^4}{4!} \right)$$

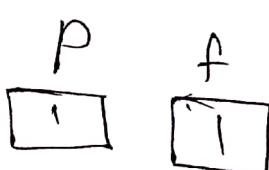
$$e(n, 2) \Rightarrow \left(1 + \frac{n}{1} + \frac{n^2}{2!} \right) + \left(\frac{n^3}{3!} \right)$$

$$e(n, 1) \Rightarrow \left(1 + \frac{n}{1} \right) + \left(\frac{n^2}{2!} \right)$$

$$e(n, 0) = 1 + \frac{n}{1}$$

$$P \leftarrow P * n \quad f \leftarrow f * 1$$

Static variable



$n = n * n \quad f = f * 1$

`int fce(int n, int n)`

static int p=1, f=1;

inf r^a;

if ($n = 0$)

return 1;

else

~~scribble~~

$$\{p \in e(n, n-1) ;$$

$$P = P \otimes N$$

$f = f * n;$
return $r + \frac{P_f}{f};$

$\{ \text{ } + \text{ } \times \text{ } \}$ \rightarrow setup

3

class-24

Taylor series e^n

$$\text{Get this, } e^n = 1 + \frac{n}{1} + \frac{n^2}{2!} + \frac{n^3}{3!} + \frac{n^4}{4!} + \dots + n \text{ terms}$$

mult this by $n!$ to get $O(n!)$

$$\frac{n!}{1*2} = \frac{n!}{1*2*3} = \frac{3}{3} = \frac{6}{6} = \frac{8}{8}$$

($n-1$) numbers

$$2[1+2+3+4+\dots]$$

$$(2+n)^n + 1 = 2 = 2 \frac{n(n+1)}{2} \text{ (2 numbers)}$$

$$((1+n)^n)^2 \text{ (n numbers)} = n(n+1) \Rightarrow O(n^2)$$

multi

$$\Rightarrow 1 + \frac{n}{1} + \frac{n^2}{1*2} + \frac{n^3}{1*2*3} + \frac{n^4}{1*2*3*4}$$

$$\Rightarrow 1 + \frac{n}{1} \left[1 + \frac{n}{2} + \frac{n^2}{2*3} + \frac{n^3}{2*3*4} \right]$$

$$\Rightarrow 1 + \frac{n}{1} \cdot \left[1 + \frac{n}{2} \left[1 + \frac{n}{3} + \frac{n^2}{3*4} \right] \right]$$

$$\Rightarrow 1 + \frac{n}{1} \left[1 + \frac{n}{2} \left[1 + \frac{n}{3} \left[1 + \frac{n}{4} \right] \right] \right]$$

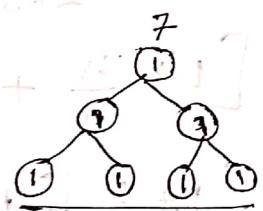
↑ ↑ ↑ 3 → multi

$O(n)$

nlogn

```
int e(int x, int n)
{
    int s = 1;
    for (; n > 0; n--)
        s = 1 + w_n * s;
    return s;
}
```

```
int e(int n, int n)
{
    static int s = 1;
    if (n == 0)
        return s;
    s = 1 + w_n * s;
    return e(n, n-1);
}
```



$$\begin{array}{r} 2 \ 3 \ 4 \ 5 \ 6 \\ \swarrow \searrow \end{array} \quad \begin{array}{r} 3 \ 4 \ 5 \\ \swarrow \searrow \end{array} \quad \begin{array}{r} 4 \ 5 \\ \swarrow \searrow \end{array} \quad \begin{array}{r} 5 \ 6 \\ \swarrow \searrow \end{array}$$

2 3 4 5 6 7 8 9 10 11 12

13 14 15 16 17 18

Fibonacci Series

Class - 27

$\text{fib}(n)$	0	1	1	2	3	5	8	13
n	0	1	2	3	4	5	6	7

$$\text{fib}(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ \text{fib}(n-2) + \text{fib}(n-1) & n > 1 \end{cases}$$

int fib(int n)

```
{ if (n <= 1)
    return n;
```

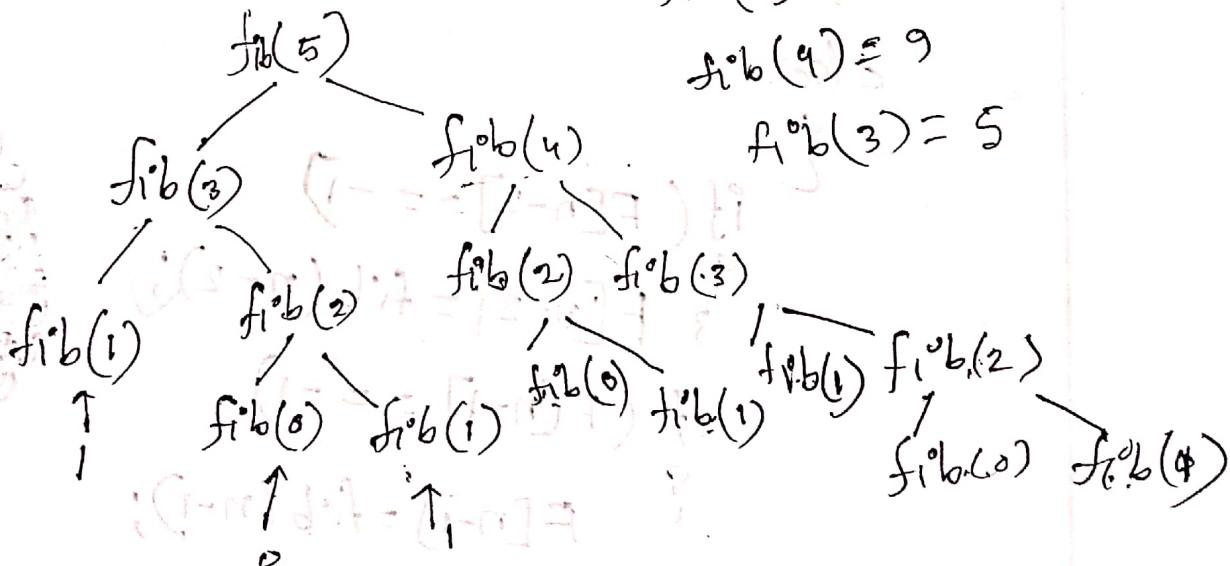
```
return fib(n-2) + fib(n-1); } // Time Complexity O(2^n)
```

}

$\text{fib}(5) = 15$ calls

$\text{fib}(9) = 9$

$\text{fib}(3) = 5$



$(i-n) + (i-n)$ numbers

Recursive Function

if the Recursion function ~~helps~~ the call the same value many times then if called the Excessive Recursion

Using Memosization

```
int F[10];  
if  
int fib(int n){  
    if (n <= 1) {  
        F[n] = n;  
        return n;  
    }  
    else {  
        if (F[n-2] == -1) {  
            F[n-2] = fib(n-2);  
        }  
        if (F[n-1] == -1) {  
            F[n-1] = fib(n-1);  
        }  
        return F[n-2] + F[n-1];  
    }  
}
```

combination Formula

Class-29

$$nC_r = \frac{n!}{r!(n-r)!}$$

selection

$$C \rightarrow {}^5C_0, {}^5C_1, {}^5C_5$$

0-5

A B C D (E F G H I J K L M)

select 3

A B C

A B D

A C B

01789116343

int C (int n, int r)

{
 int f1, f2, f3;

f1 = fact(n);

f2 = fact(r);

f3 = fact(n-r);

return f1 / (f2 * f3);

b
hosseinsabbir247
@gmail.com

82
sabb 1

~~sabb 2~~

Sabb 4321

3

O(n)

sabb 4321

Combination Formula

$${}^n C_r = \frac{n!}{r!(n-r)!}$$

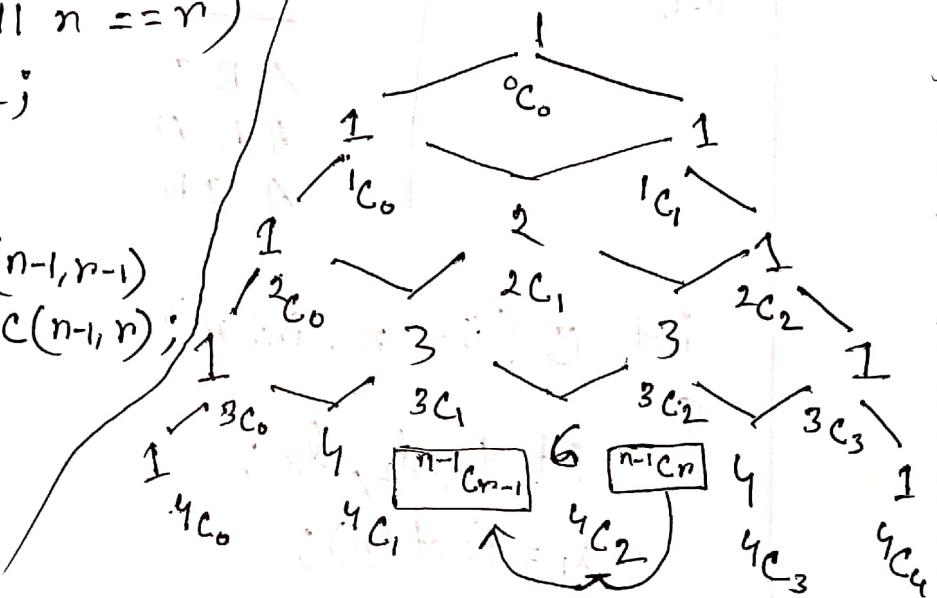
int C(int n, int r)

```
{ if (n==0 || n == r)
    return 1;
```

```
else
```

```
return C(n-1, r-1)
    + C(n-1, r);
```

Pascals triangle



(n-1, r-1) highlighted
(n-1, r) highlighted
(n-1, r) highlighted