Sub :

Day

Time : Date : / /

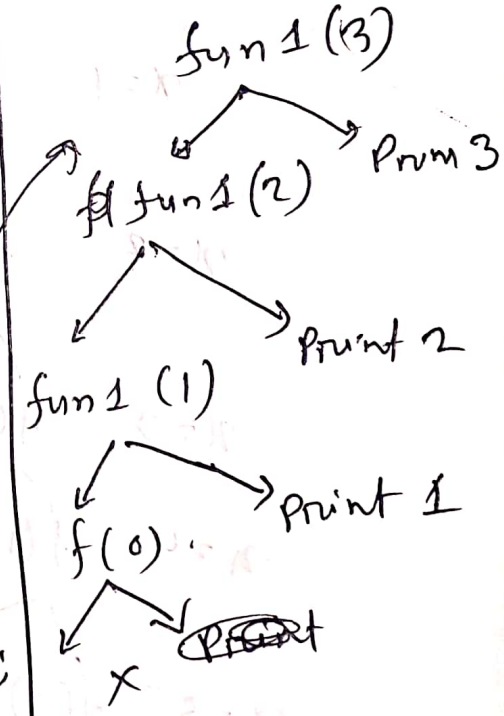
Recursion

```
Void func1 (int n)
{
    if (n>0)
    { print ("%d", n);
      fun (n-1);
    }
}
Void Main()
{
    int x = 3;
    fun1(x);
}
```

Output : 3, 2, 1.

```
void fun1 (int n)
{
    if (n>0)
      fun1 (n-1);
    printf ("%d", n);
}
```

output :- 1, 2, 3



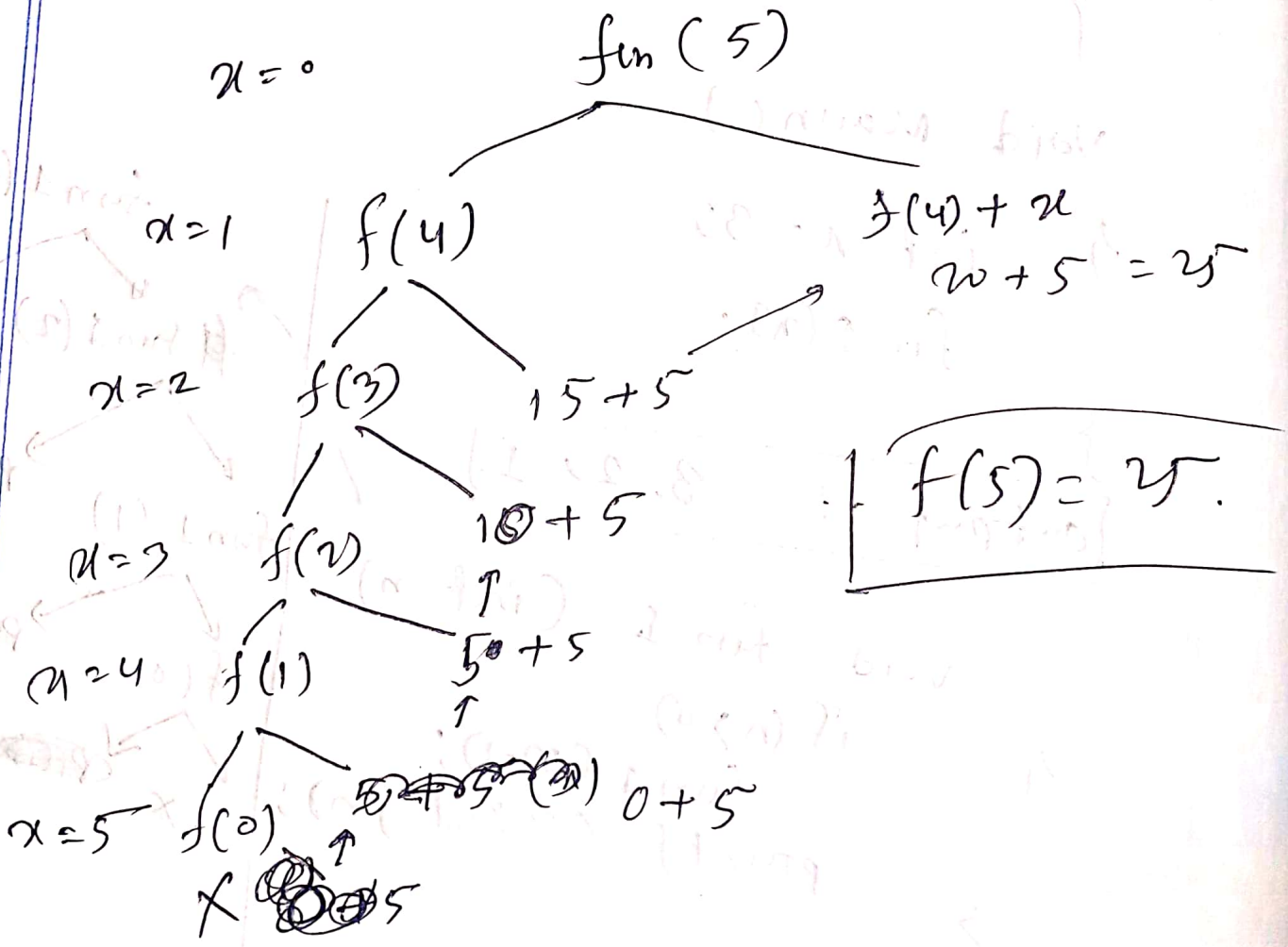
Sub :

Day

Time : Date : / /

Static Variable

```
int fun.(int n)
static int x = 0;
if (n > 0)
{ x++;
} return fun(n-1) + x;
```



$f(5) = 25$.

Sub :

Day

Time : Date : / /

Types of Recursion

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

1. Tail recursion (Last statement)

→ Same as loop.

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

Sub :

Day
Time :
Date : / /

2. Head recursion:

not same as loop

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

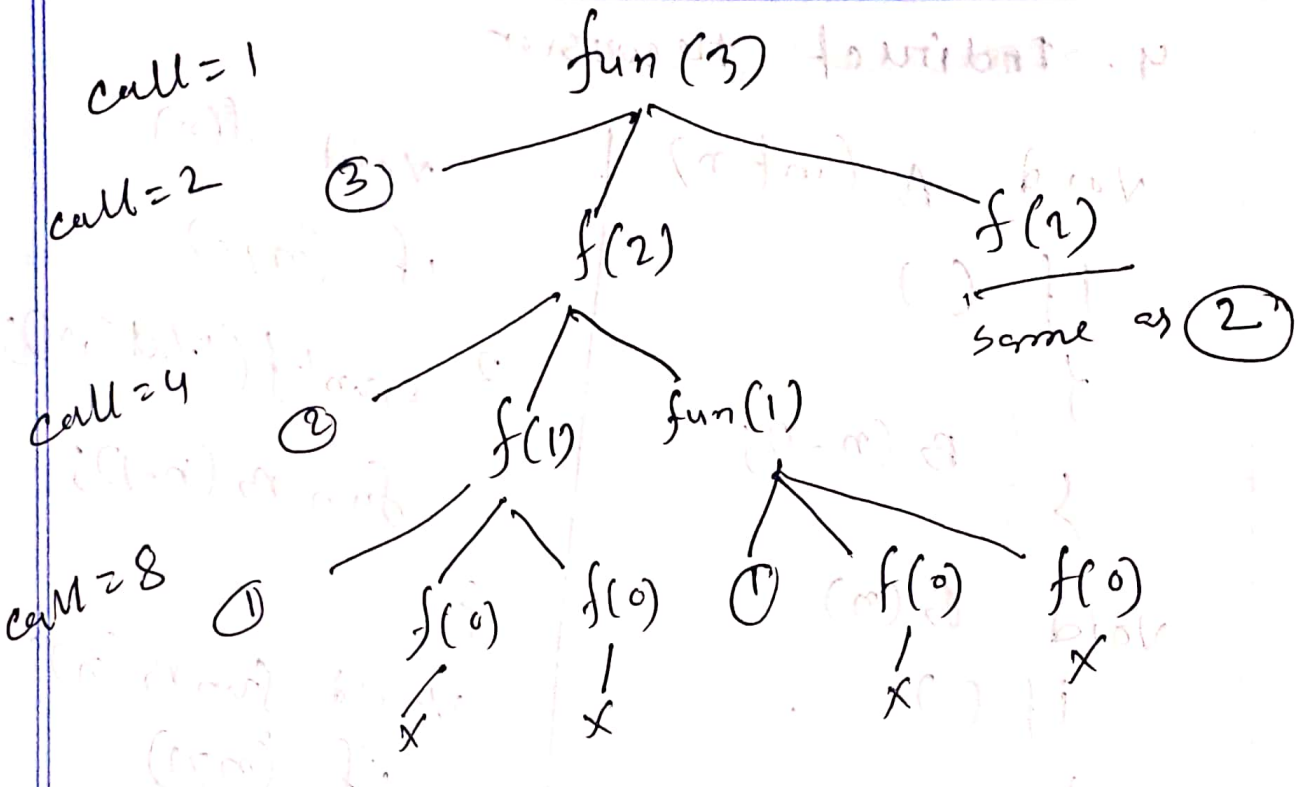
4. Tree recursion (more than 1 time)

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

Sub :

Day

Time : Date : / /



3 2 1 1 2 1 1

total call = 1 + 2 + 4 + 8 = 15.

$= 2^{n+1} - 1$

$\approx O(n^n)$

Sub :

Day

Time : Date : / /

4. Indirect recursion

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

```
void f(n)
if (n>0)
{ printf("%d", n);
  fun B (n-1);
}
void fun B (n)
  if (n>1)
{ printf("%d", n);
  fun A (n/2);
}
```

5. Nested recursion

```
int fun (int n)
{ if (n>100)
    return n-10;
  else return fun ((fun (n+11)).
```

Sub :

Day

Time : Date : / /

Use of recursion

1. Sum of Natural Number

Sum (n) - 1+2+3 ... (n-1) + n.

Sum = Sum (n-1) + n

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

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

2. factorial

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

```
int f (n)
  if (n == 0)
    return 1.
  else
    return f(n-1) * n.
```

complexity

time : O(n)

space : O(n)

Sub :

Day

Time : Date : / /

3. Power

$m^n = m * m * \ldots n$ time

pow (m, n) = (m * * ... n-1) * m

Po(m, n) = pow (m, n-1) * m

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

4. Fibonacci

time (Log n)

2^n

space O(n)

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

```
int fib (int n)
    if (n <= 1)
        return n;
    else, return f(n-2) + f(n-1)
```

Sub :

Day

Time : Date : / /

Taylor series $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} \cdots$

```
int e (int x, int n)
    static int p=1, f=1;
    int r;
if (n==0)
    return 1;
else
{   r = e(x, n-1);
    p = p*x;        → static.
    f = f*n;
    return r + p/f
```

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

Sub :

Time : Date : / /

combination

```
int c (int n, int r)
{
    a = fact(n);
    b = fact(r);
    c = fact(n-r);
    return a/b*c
}
```

finding any term:

```
int c (int n; int r)
    if (r2=0 || n=r)
        return 1;
    else
        return C(n-1, r-1) + c(n-1,r
```