Lecture :- Recursion 1

Agenda

- Basics
- Recursive codes
- Recursion call tracing

TC
SC } - Next class

Class starts at 8:35 PM

Why recursion?

- Trees
- Dynamic programming
- Graphs
- Backtracking

Recursion

1. function calling itself.
2. Solving a problem using subproblems.

Example:

$$sum(n) = 1 + 2 + 3 + 4 + 5 \cdots\cdots n.$$

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

$$sum(n) = sum(n-2) + n-1 + n.$$

$$sum(6) = 1 + 2 + 3 + 4 + 5 + 6 = 21$$

$$sum(6) = sum(5) + 6$$

$$= [1 + 2 + 3 + 4 + 5] + 6 = 21.$$

How to write recursive codes?

Three steps:

1. Assumption: what your function should do?
2. Main logic
3. Condition to stop recursion: Base case

Qu Given a no n, find and return 1 + 2 + 3 ------ n

using recursion.

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

1. Assumption

Given a no. n, return sum of first n natural numbers.

2. Main logic

$$\begin{tikzcd} & \text{sum}(n) \arrow[d] \\ \text{smallAns} = & \text{sum}(n-1) \arrow[u, bend right, "\text{smallAns} + n."'] \end{tikzcd}$$

3. Base case

Solve the smallest subproblem.

which can't be broken.

function call tracing

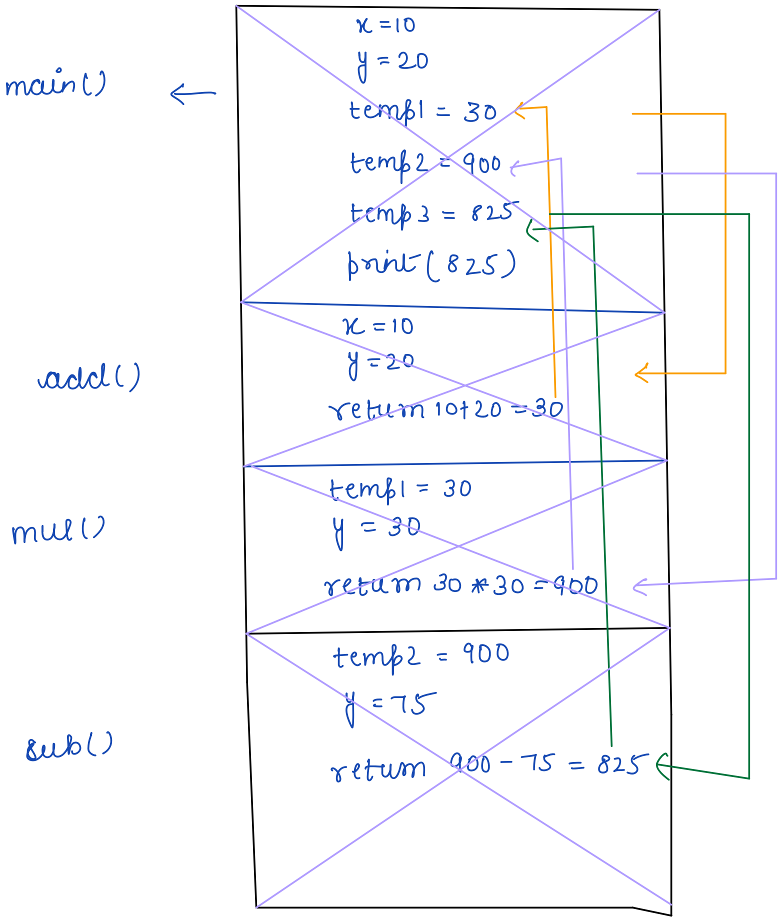
[function stored in stack memory]

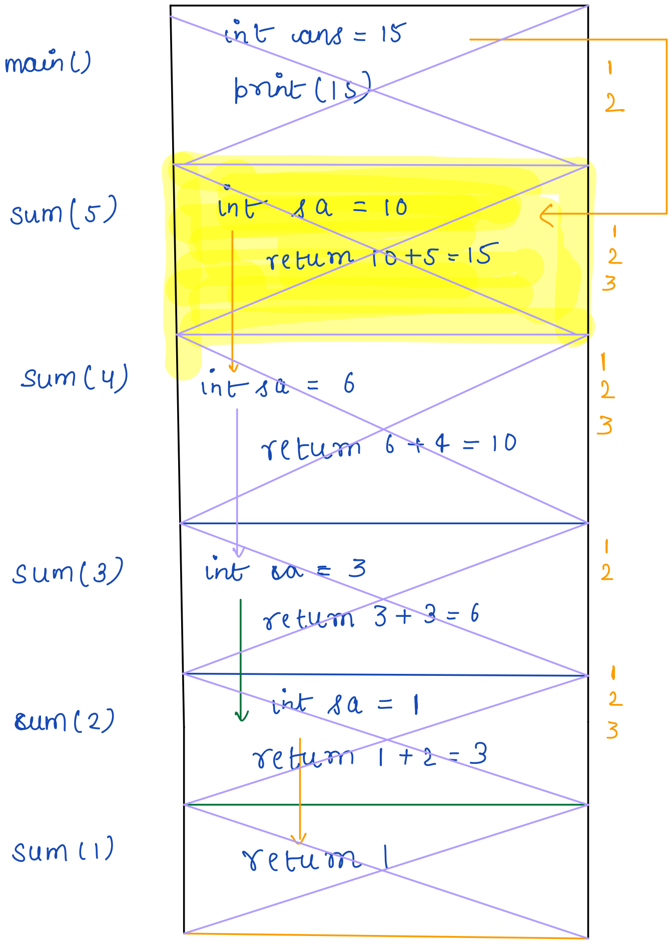
function inside a stack

takes O(1) space

```
main() {
1. int x = 10;
2. int y = 20;
3 int temp1 = add(x,y);
4 int temp2 = mul( temp1, 30);
5 int temp3 = sub ( temp2, 75);
6 print (temp3);
}
```

```
int add(int x, int y){
    return x+y;
}

int mul (int x, int y){
    return x *y;
}

int sub (int x, int y){
    return x -y;
}
```





```
        int sum(int n) {
1.          if(n==1) {
               return 1;
            }
2.  int sa  = sum(n-1);
3.  return sa + n;
    }

    main() {
    1  int ans = sum(5);
    2  print(ans);
}
```

Qu find factorial of a number

fact(4) = 4 * 3 * 2 * 1 = 24.

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

    return sa * n;
}
```

Assumption

Given a number n.
find and return
factorial of n.

Main logic

fact(5) = 1 * 2 * 3 * 4 * 5

fact(5) = fact(4) * 5

$$\begin{tikzcd}
\text{fact}(n) \arrow[d] \\
sa = \text{fact}(n-1) \arrow[u, bend right, "sa * n."']
\end{tikzcd}$$

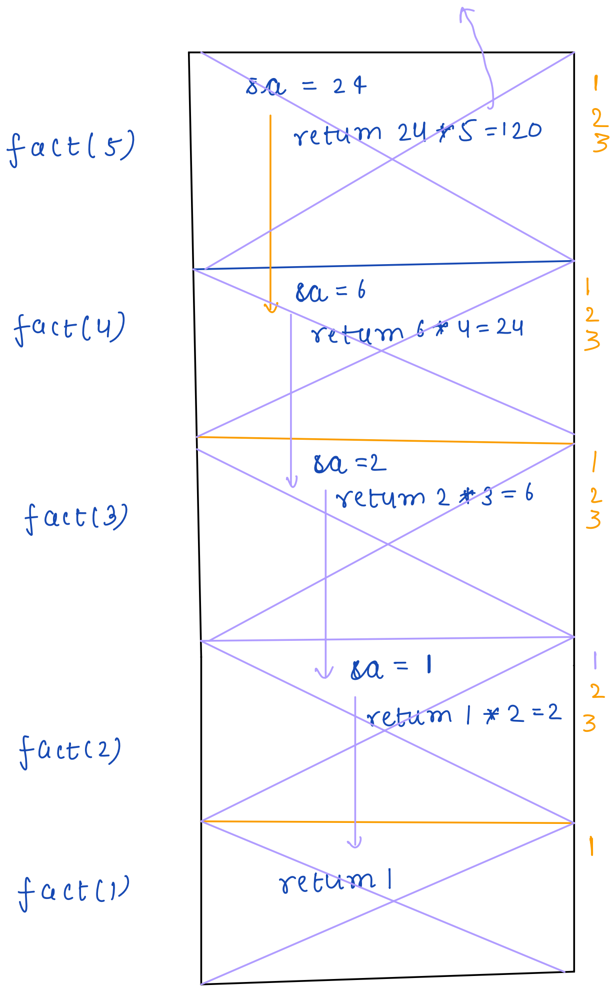
Base case

fact(1) = return 1.

fact(0) = return 1.

1 <= n <= 1000 [Base case: n==1]

0<=n<=1000 { Base case: n==0]



```
int fact(int n) {
 1  if(n==0|| n==1){
        return1;
    }
2 int sa = fact(n-1);

3. return sa*n;
}
```

Break: 9:45 PM

Qu Print nth fibonacci number.

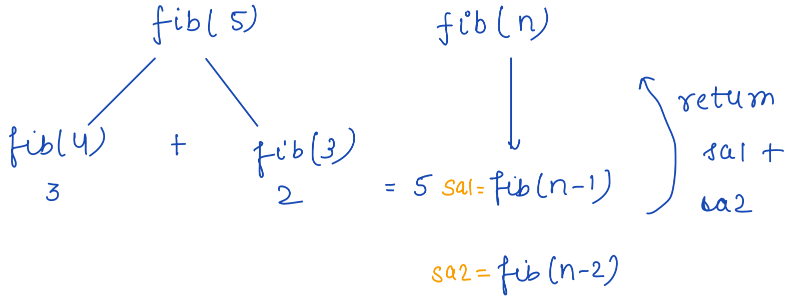
0	1	1	2	3	5	8	13	21	34 ⇒ 21 + 13 55-
0th	1st	2nd	3rd	4th	5th	- - - -			

```
int fib(int n) {
        if(n==0 || n==1){
            return n;
        }
      int sa1 = fib(n-1);

      int sa2 = fib(n-2);

      return sa1 + sa2;
}
```

Assumption

Given a number n, find and return nth fibonacci number.

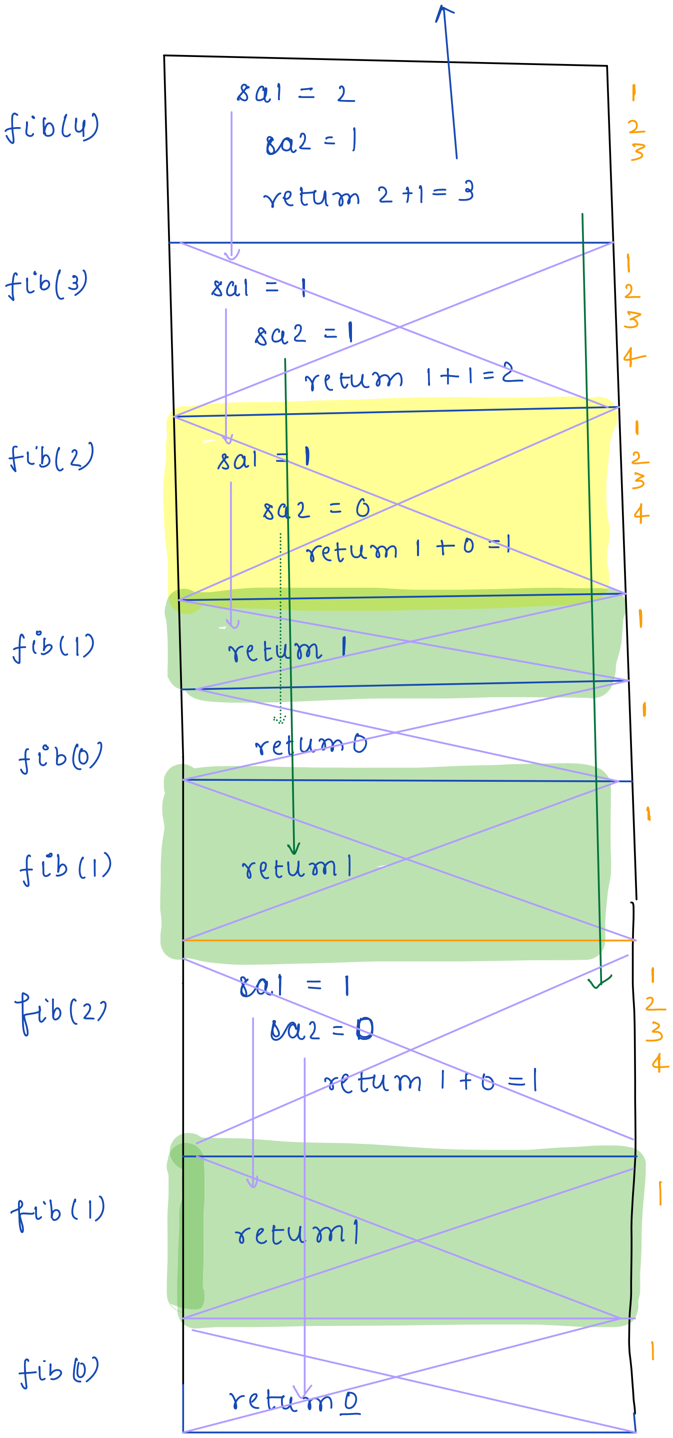
Main logic



Base case

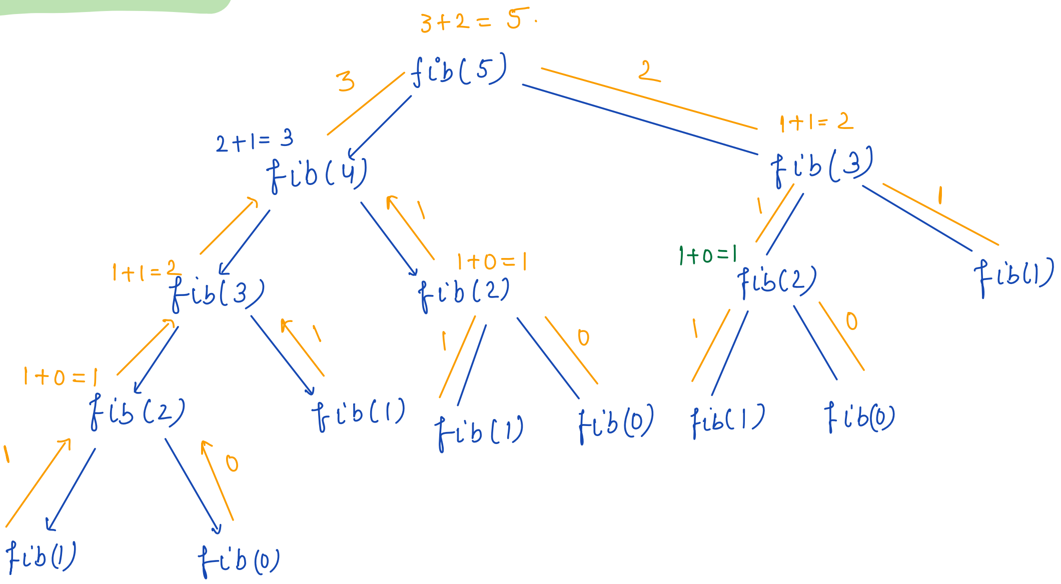
fib(1) = 1

fib(0) = 0

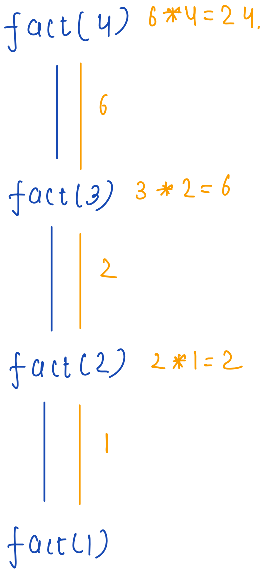


```
int fib(int n) {
1     if(n==0 || n==1){
           return n;
      }
2     int sa1 = fib(n-1);

3     int sa2 = fib(n-2);

4     return sa1 + sa2;
}
```

Recursion tree



factorial



Qu. Given a ch[], check if its a palindrome or not?

```
boolean isPalindrome(char[] ch, int s, int e) {
        if (s >= e) {
                return true;
        }
      if (ch[s] == ch[e]) {
          return isPalindrome(
                        ch, s+1, e-1);
       }

      return false;
}
```

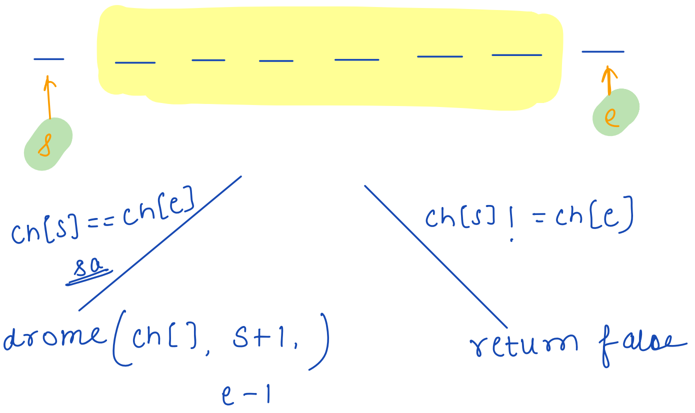
Assumption

Given a ch[], s, e.

whether a ch[] from s to e

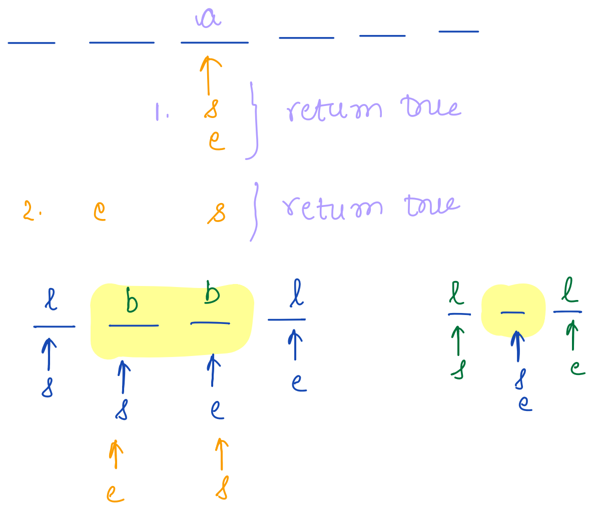
is a palindrome?

Main logic



H|w: Dry run. (h|w)

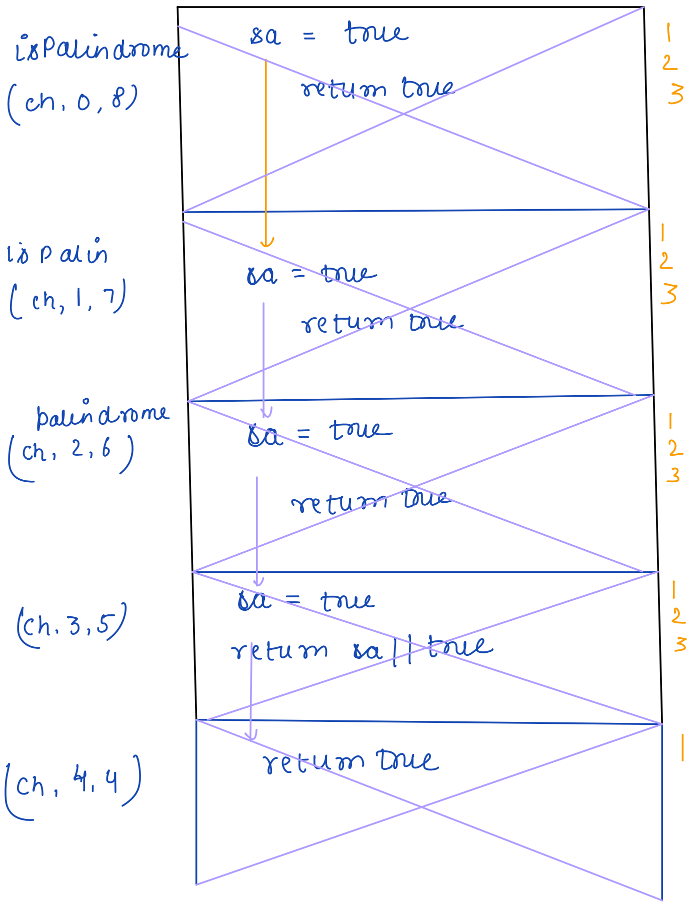
Base case



Dry run:

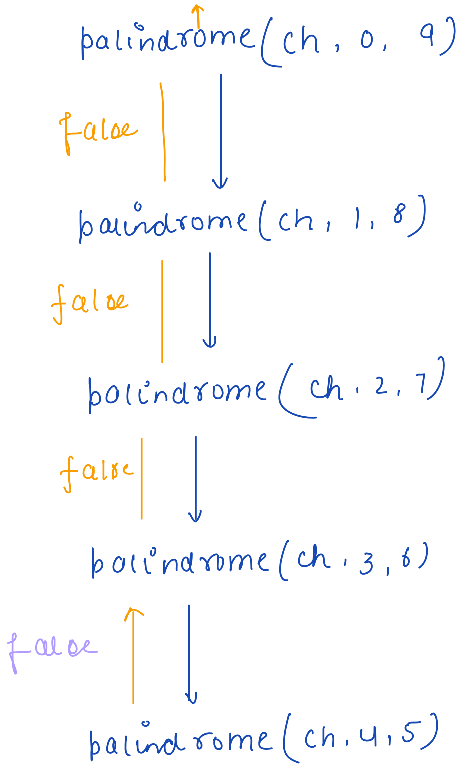
	0	1	2	3	4	5	6	7	8
ch[] =	m	a	l	a	y	a	l	a	m.

s = 0
e = 8



```
boolean isPalindrome(char[] ch, int s, int e) {
  1  if (s >= e) {
            return true;
     }
 2 if (ch[s] == ch[e]) {
   3  sa = isPalindrome(
                ch, s+1, e-1);
     } return sa;

 4 return false;
}
```

m a l a y k a l a m
0 1 2 3 4 5 6 7 8 9



palindrome(ch, 0, 9)

false

palindrome(ch, 1, 8)

false

polindrome(ch, 2, 7)

false

polindrome(ch, 3, 6)

false

palindrome(ch, 4, 5)

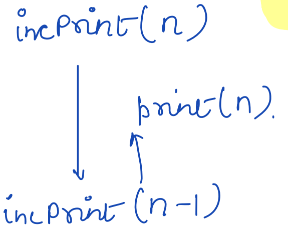
Assignment: print numbers from 1 to n in increasing manner.

```
void incPrint( int n) {
        if (n==1) {
            print(1);
            return;
        }
      incPrint(n-1);
      print(n);
}
```

Assumption

Main logic

1 2 3 4 5 6 7



Base case

n==1 : print(1)
return.

Thankyou ☺