1

Recursion

I.
```
int factorial (int n) {
// base case
if (n==0)
   return 1;
int chhoti = factorial (n-1);
int badi = n * chhoti;
return badi;
}
int main() {
 int n;
 cin>>n;

int ans = factorial(n);
cout << ans << endl;

}
```

[illegible] code [illegible] nhi

```
int factorial (int n) {
 if (n==0)
   return 1;
 return n * factorial(n-1);
}
 int main() {
  int n
  cin >>n;
  int ans = factorial(n);
   cout << ans << endl;

 }
```

10 int smaller Problem = factorial (n-1);
11 int bigger problem = n * smaller problem;

1x 1 = 1

1 base case
2 procesing
3 recursive call

→ V.V.g

Q 2. fabonacci

0, 1, 1, 2, 3, 5, 8, 13, 21

1) $f(n) = f(n-1) + f(n-2)$ → recursive call

2) Base case
```
↳ if (n==0)
     return 0;
  if (n==1)
     return 1;
```

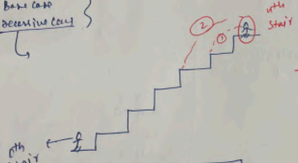
idea

PMI

1) P(0) / P(1) → true
   1 case solve karna
2) f(k) → true
3) f(k+1) → true

3. Count ways to reach N-th stairs

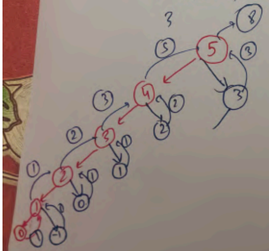
① Base case
② recursive case }
↳



Condition
↳ 1step
↳ 2step.

```
R.R → f(n) = f(n-1) + f(n-2)

B.C → if (n<0)
        return 0;
      if(n==0)
      { return1;
      }
```



4. Say digits

String arr[10] = {"zero", "one" - - - "nine"}

432 → digit
↳ int digit = num%10
↳ num = num/10;

```
① void say digit (int n, string arr[i]) {
   // say digit (412, arr)
Baso case // ② if (n==0)
   ③ return;
   ④ int digit = n%10;
   ⑤ n = n/10;
   ⑥ say digit (n, arr);
      ↳ say digit (41, arr)
        ↳ say digit (4, arr)
          ↳ say digit (0, arr);
0
```

digit = 412%10 = 2 =

n = 41

digit = 41%10 = ①

n = 4

digit = 4%10 = 4

n = 0

L-33

Recursion | Binary search - 03

5. is Sorted Array ? [2 | 4 | 6 | 9 | 11 | 13]

arr[0] arr[1]

Base case →

```
    if (size == 0 || size == 1)
        return true;
```

process

```
  if (arr [0] > arr [1])
      return false;
  else
      bool ans = is sorted (arr+1 ; size -1)
      return ans;
```

Q6 i/p → [3 | 2 | 5 | 1 | 6]

O/P → sum → 17

```
Base case →  if ~~sum~~(size == 0 )
                 return 0;
             if (size == 1 )
                 return arr[0];
      int remaining part = get sum(arr+1, size -1)
      int sum = arr[0] + remaining part;
  return sum;
```

Q7 Linear search

array → [1 | 2 | 3 | 4 | 5]

key | element = 6

O/P - found | not found.

Base case

```
    if (size == 0)
        return false;
    if ( arr[0] == k){
        return true;
    }
    else{
        bool remaining part = linear search(arr+1, size-1, k);
        return remaining part;
    }
```

[0 | →]

k

Q8. Binary search

```
bool binary search (int * arr, int s, int e, int k) {
  // base case                  +   if (arr [mid] == k)
  if ( s>e)                             return true;
    return -1;
    int mid = s + (e - s ) /2) ;
  if (arr[mid] < k ){
    return binary search( arr, mid+1, e, k);
  }
  else
    return binary search (arr, s, mid-1, k );
}
```

Recursion with string

Q9. i/p = "kuvlay"   reverse (i, j, s)

o/p = " yalvuk"

```
Base case //
        if ( i > j )
        return;
    swap ( s[i], s[j] )
      i++ ;
      j-- ;
Recursive call
    reverse ( str, i, j );
  }

String reverse String (String str)
{
 reverse ( str, 0, str.length() -1 );
 ~~reve~~return str;
 }
```

Q10   **Palindrome**

(1) → String = abba

~~Dar~~

abbc cbba
i   j

→ O(n) T.C

→ O(1) S.C

Palindrome

```
// base case
→ if ( i>j )
    return;
  if( str[i] != str[j])
    return false;
 else
→ // recursive case
  return check palindrome( str, i+1, j+1);
 }
}
```

Q11. i/p a=3 , b = 2

o/p $3^2 = 9$

a^b → b is even → $a^{b/2} \times a^{b/2}$

→ b is odd → $a \times (a^{b/2} \times a^{b/2})$

```
base case //  if ( b==0 )
                return 1;
              if ( b==1 )
                 return a;
  // Recursive call
→    int ans = power (a, b/2);
   if ( b%2 == 0 ) {
     return ans * ans;
  }
  else {
    return a * ans * ans;
  }
```

3, 11

3^5