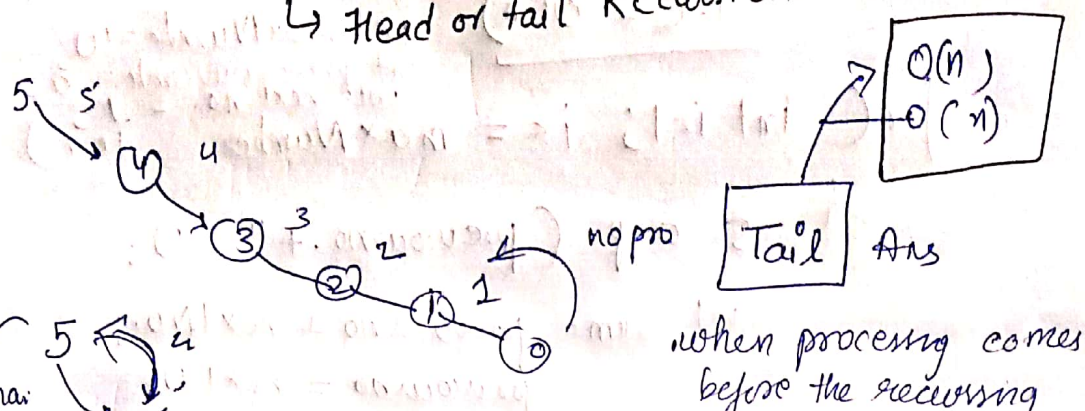
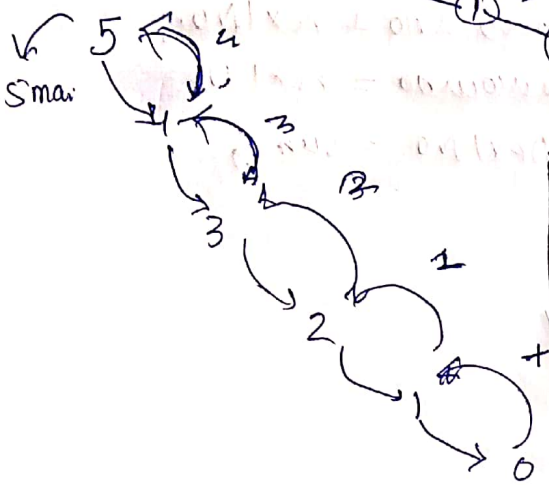
Ques 1. 5 4 3 2 1

↳ Head or tail Recursion



Ques 2



when processing comes before the recursing caling

T.C → O(n)
M.C → O(n)

Head Ans

when ~~processing~~ ~~call~~ recursing call comes before the processing

Ques:- T.C ⇒

M.C.

Ques:-
factorial

M.C → O(n)
T.C → O(n)

Ques. Fibonacci → Optimise → without recursion

Time comp O(2^n)
O(n)

Also Optimised using DP. O(n) (On) optimise

```
Pubic clan fibo {
    final int maximum = 10000;
    int fib[] = new int [maximum];
    int fibonacci( in num){   if (num == 0) return 0;
                              if (num == 1) return 1;
```

Optimize

Using for loop fibonnacci S.C = O(1)

T.C = O(n)

```
int maxNumber = 10;
int previousNumber = 0;
int nextNo. = 1;
for( int i=1; i<= maxNumber ; i++)
{
    S.O.P-(previousno.+" ");
    int sum = previousno + nextNo.;
    previousno = nextNo;
    NextNo. = sum;
}
```

Ques:

Print the Number in words

T.C = O(n)

S.C = O(n)

Ques.

~~1) Exponent~~

T.C: O(n)

S.C: O(n)

2) ~~Sorted or Unsorted~~ Sorted unsorted

S.C: O(N)

T.C: O(N).

Ques.. Subset using Bitmasking

Pseudocode

```
void subsets ( int arr[] , int n)
{
    int subset_size = pow(2,n)
    int index, i.
    for( index from 0 to subset_size)
    { int subset[n]
      for( i from 0 to n)
      { if (index & i<<i ))
        { subset[i] = arr[i]
```

print(subset)
}

T.C ⇒ O(n * 2 ^ n)
M.C ⇒ O(n)

Total way to reach nth stair from bottom
T.C ⇒ O(3^n) [because 3 recursive function are called]
S.C ⇒ O(1) → No sapace are require.

9) Subsequence.
T.C :- O(2^n)
S.C :- O(n)
optimised using Dynamic Programming

10) Permutation
T.C :- O(n * n!)
S.C : O(n)

11) source to Destination [backtracking is required because we want previous result]

Pseudo code
Let destion p, q.
Let origin x, & y
let m is row n is column.

// To move right.
if (move(x+1, y, m, n){
output.push_back('R');
print(m, n, x+1, y, p, q, output);
output.pop_back();

// To move up.
if (move, x, y+1, m, n){
output.push_back('U');
print (m, n, y+1, p, q, output);
output.pop_back();
}