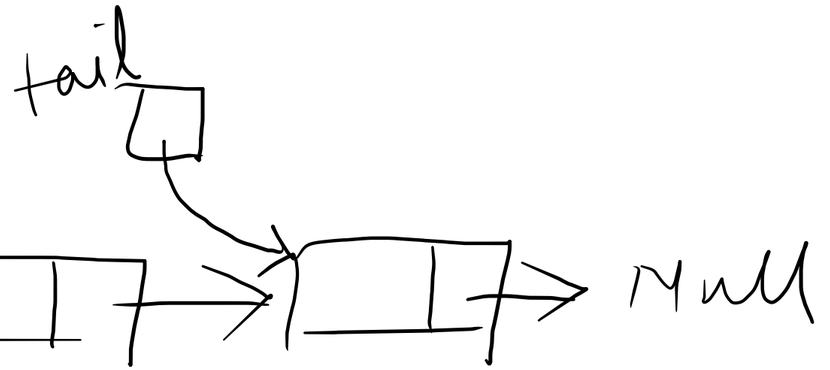
Linked List

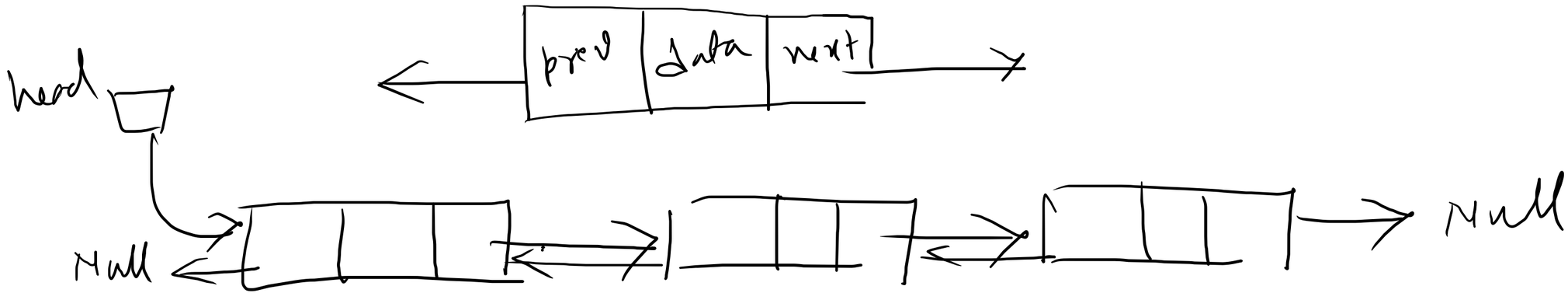
1) Singly linked list.

variations of linked list

2) singly linked list with tail pointer.



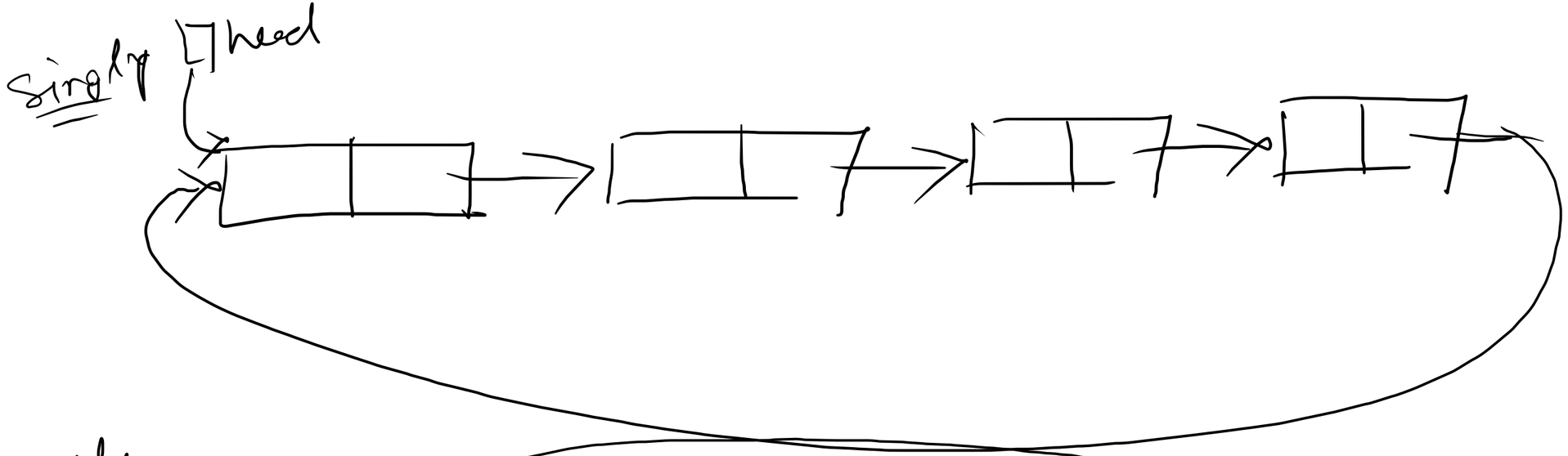
3) Doubly linked List.



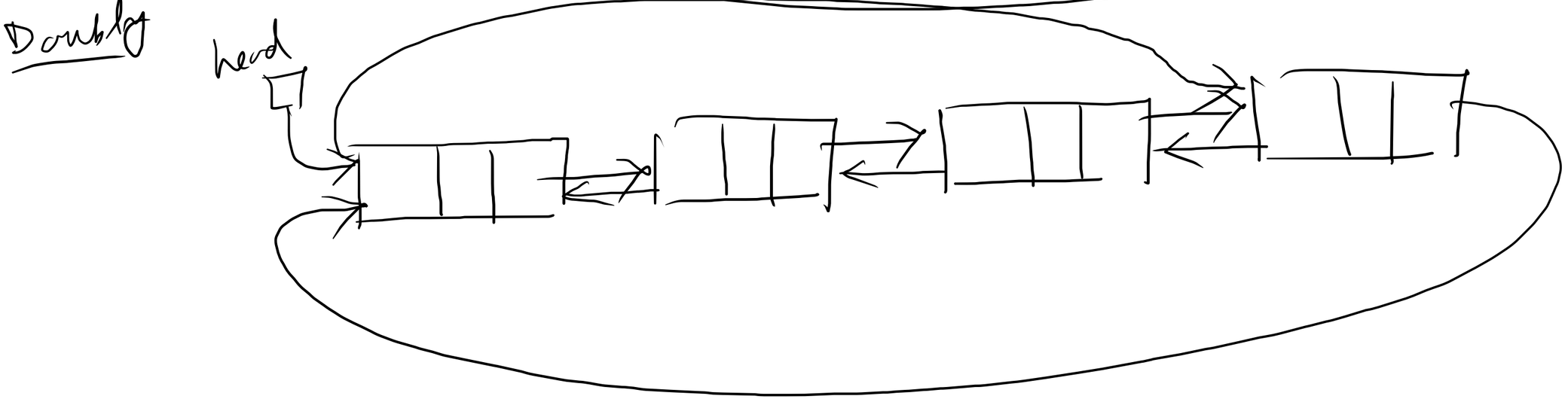
4) Doubly linked list with tail pointer.

5) circular linked list

Singly



Doubly



H.W 1> Merging one doubly linked list at the end of another.

1. without tail pointer
2. with tail pointer.

2> converting a singly linked list into a dobly linked list.

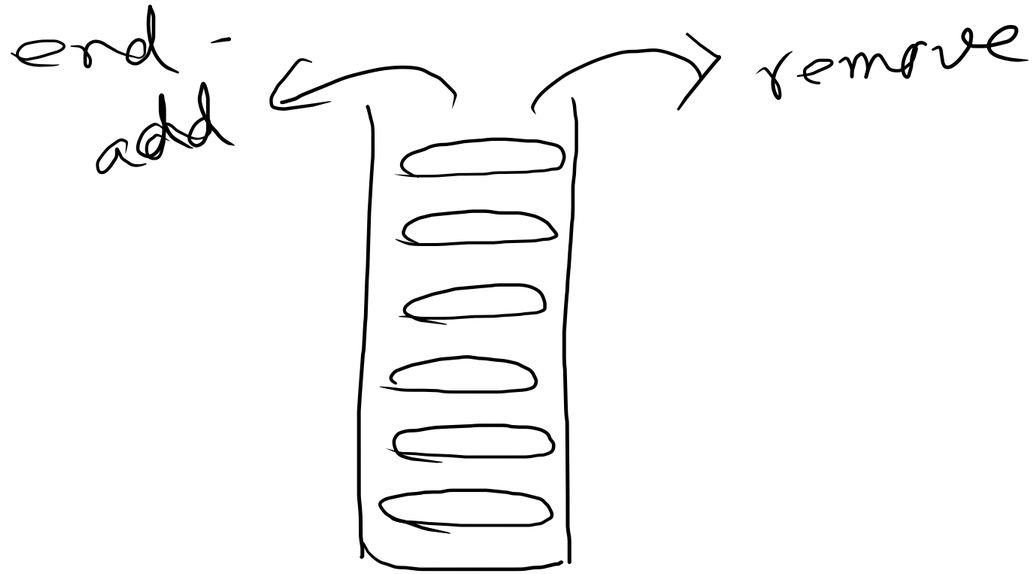
3> Remove the first element and add at the end.

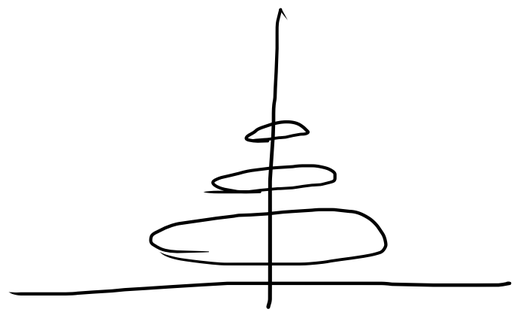
i) singly
ii) doubly.

4> selection sort using linked list.

Stack data structure

- Insert and delete can be performed at the same end -





Tower of Hanoi

LIFO ← last in first out

Application i) function call
ii) undo operation / actions in web browsers.

Stack implementation

Two ways : array based

Linked-list based.

Operations

dynamic

Push (K)

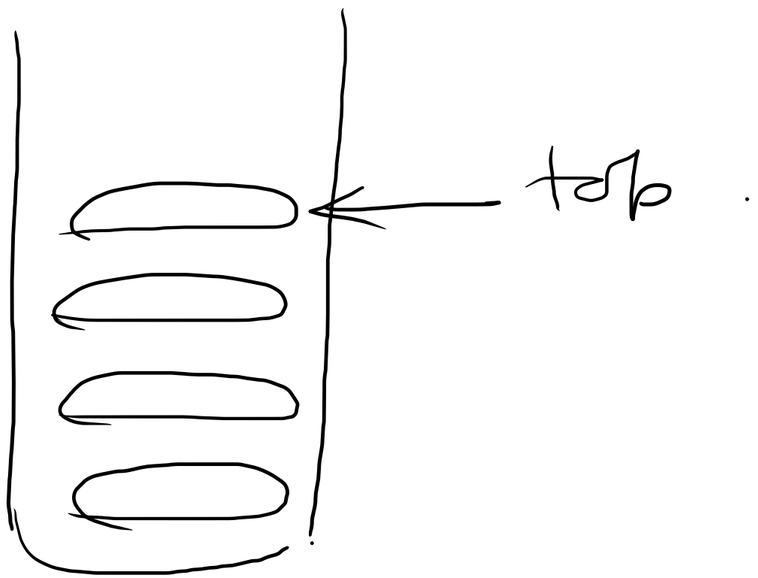
Pop ( )

Static

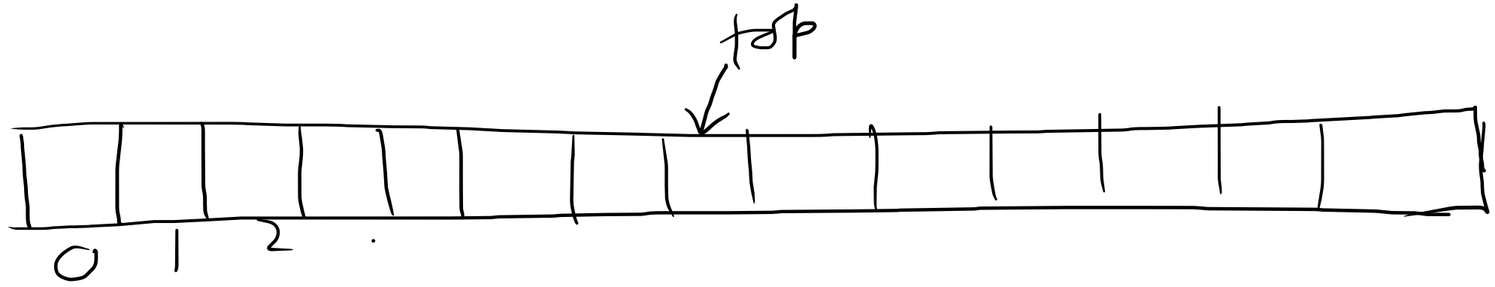
Isfull ( )

Isempty ( )

top ( )



Array - based



0 - top : elements of stack are present.

Size
capacity.

- no element in the stack : top = -1

```
IsFull(A)
{
    if top = size - 1
        return true
    else
        return false
}
```

```
IsEmpty(A)
    if top == -1
        return true
    else
        return false
```

```
Top(A)
    return top
```

```
Push (A, k)
  If  Is full (A)
        Insert not possible
  else
        top = top + 1
        A [top] = k
```

$\theta(1)$

```
Pop (A)
  If  Is empty (A)
        Deletion not possible
  else
        top = top - 1
```

$\theta(1)$

Linked-list based

1) i) Insert-at-beg — Push head can be visualize as top.
2) ii) Delete-at-beg — Pop.

Implement one application of stack

i) Evaluation of an expression
ii) conversion of an expression.

3 types of expressions.
- Infix
- Prefix
- Postfix.

Infix: $x + y$ <operand> <operator> <operand>

Prefix: $* 3 5$ <operator> <operand> <operand>

Postfix $x 5 -$ <operand> <operand> <operator>

Infix expression

$2 + 3 * 5 = 17$

- Precedence
- associativity.