3). Deletion and Traversing :-

① Deletian at beginning :- Removing the node from beginning of the list.

②. Deletion at end :- Remoring the node from end of the list.

Traversing :- visiting each node of the list atleast once in order to perform some specific operation like searching, sorting, display etc.

Searching :- compairing each node data with the item to be searched and return location of the item in the list if the item found else return null.

Skip list :-

* What is a skip list ?

A skip list is a probalistic data structure. The skip list is used to store a linked list of elements or data with a linked list. In one single step, it skips sereral elements of the entire list, which is why it is known as skip list.

Structure of skip list :-

skip list is built in two layers : The lowest layer and the top layer. The lowest layer of the skip list is a common sorted linked list, and the top layers of the skip list are the like an "express line" where elements are skipped.

complexity table :-

Sr.No	complexity	Average case	Worst case
1).	Access complexity	o(logn)	o(n)
2).	search comple.	o(logn)	o(n)
3).	delete comple.	o(log n)	o(n)
4).	Insert comple.	o(logn)	o(n)
5).	Space comple.	-	o(nlogn).

Basic operations and its algorithms :-

1). Insertion operation :- It is used to add new node to a particular location in a specific situation.

2). Deletion operation :- It is used to delete a node in a specific situation.

3). search operation :- The search operation is used to search a particular node in a skip list.

Algorithm of insertion operation :-

```
Insertion (L, key)
local update [0 ... max-level +1]
q = L → header
for i = L → level down to 0 do.
        while q → forward [i] → key forward [i]
update [i] = a
```

```
a = a → forward[0]
lvl = random-level()
if lvl > L → level then
for i = L → level + 1 to lvl do
    update [i] = L → header
    L → level = lvl
a = make node (lvl, key, value)
for i = 0 to level do
    a → forward [i] = update [i] → forward [i]
update [i] → forward [i] = a
```

Algorithm of deletion operation :-

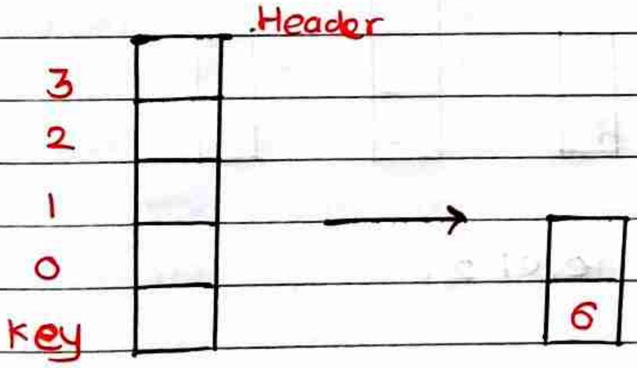
```
Deletion (L, key)
local update [0.... max level + 1]
a = L → header
for i = L → level down 0 to do.
    while a → forward [i] → key forward [i]
        update [i] = a
a = a → forward [0]
if a → key = key then
    for i = 0 to L → level do
    if update [i] → forward [i] ≠ a then break
    update [i] → forward [i] → forward [i]
free (a)
while L → level > 0 and L → header → forward [L → level]
                                    = NIL do
    L → level = L → level - 1.
```

Algorithm of searching operation :-

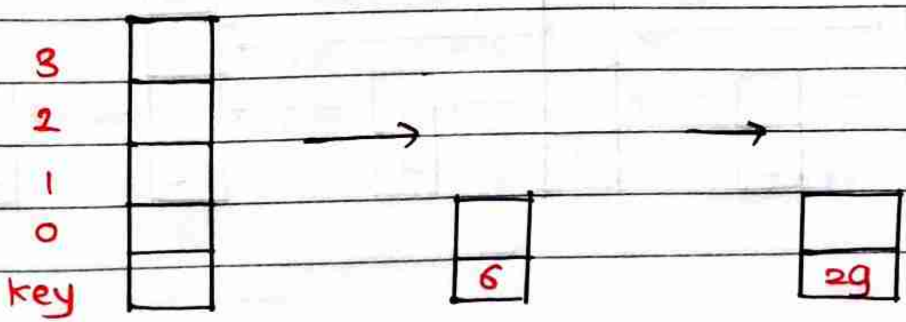
```
searching (L, Skey)
  a = L → header
  loop invariant : a → key level down to 0 do.
      while a → forward [i] → key forward [i]
  a = a → forward [a]
  if a → key = skey then return a → value
  else return failure.
```

Example : create a skip list, we want to insert these following keys in empty skip list.
1. 6 with level 1
2. 29 with level 1
3. 22 with level 4.
4. 9 with level 3.
5. 17 with level. 1.
6. 4 with level 2.

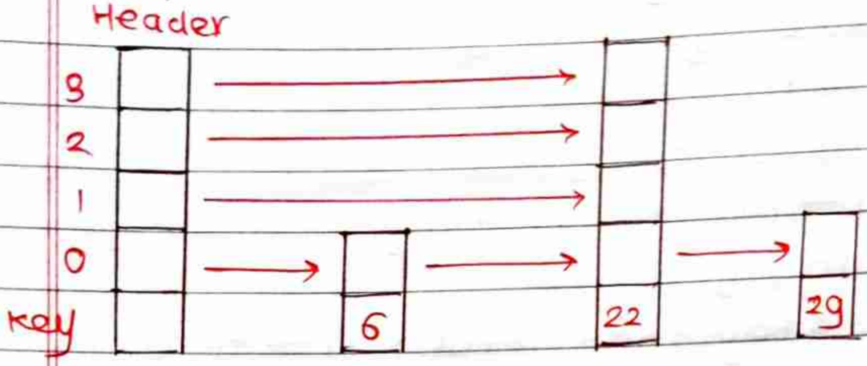
→ Solution :- Insert 6 with level 1.



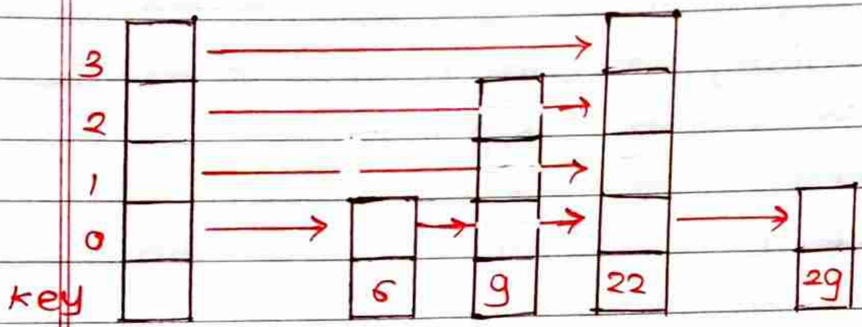
step 2 :- Insert 29 with level 1.



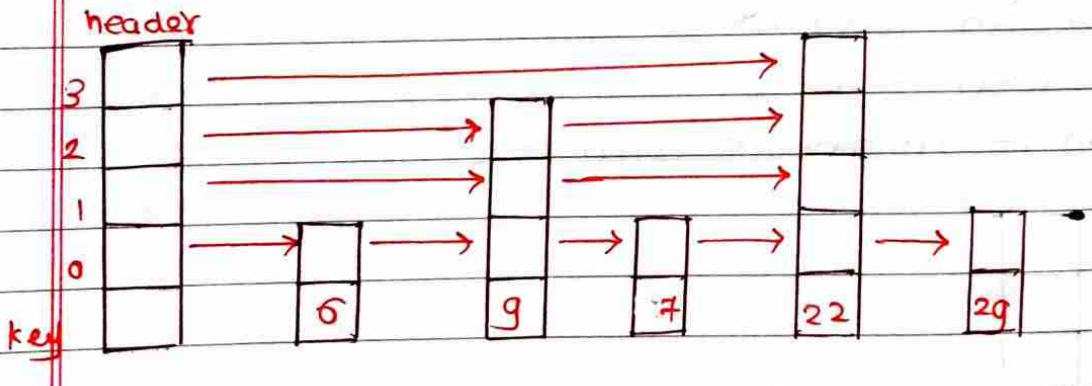
Step 3 : Insert 22 with level 4.



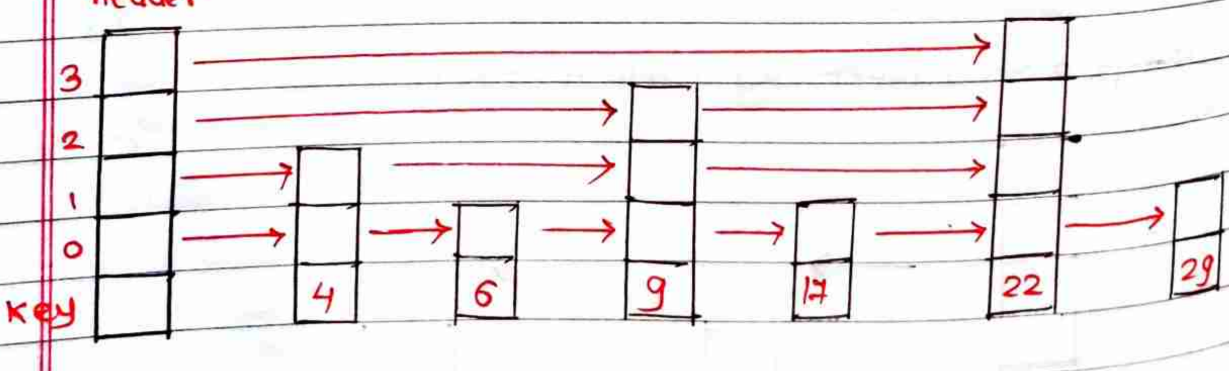
Step 4: Insert 9 with level 3.



Step 5: Insert 17 with level 1



Step 6: Insert 4 with level 2.


header
3
2
1
0
key
4
6
9
17
22
29