30-9-20 Write a prg to perform insertion, deletion & searching on a skip list. Consider the max no of levels to be log n where "n" is the no of nodes in the list.

```
#include <bits/stdc++.h>
using namespace std;
class Node
 {
   public:
      int key;
     Node **forward;
     Node (int, int);
   };
 Node ::Node (int key , int level)
    {
        this -> key = key;
        forward= new Node* [level+1];
        memset(forward, 0, sizeof(Node *)*(level+1));
    };
class SkipList
  {
        int MAXLVL;
        float P;
         int level;
        Node *header;
    public:
     SkipList (int float);
     int randomLevel();
     Node *createNode(int, int);
     void insertElement(int);      void deleteElement(int);
     void displayList();           void searchElement(int);
   };
SkipList :: SkipList (int MAXLVL, float P)
 {
   this -> ~~A~~ MAXLVL = MAXLVL;
   this -> P = P;
     level = 0;
    header = new Node (-1, MAXLVL);
   };
 int SkipList :: randomLevel()
  {
    float r = (float) rand() / RAW_MAX;
     int lvl = 0;
     while (r < P && MAXLVL)
      {
        lvl ++;
        r = (float) rand() / RAND-MAX;
      }
      return lvl;
   };
```

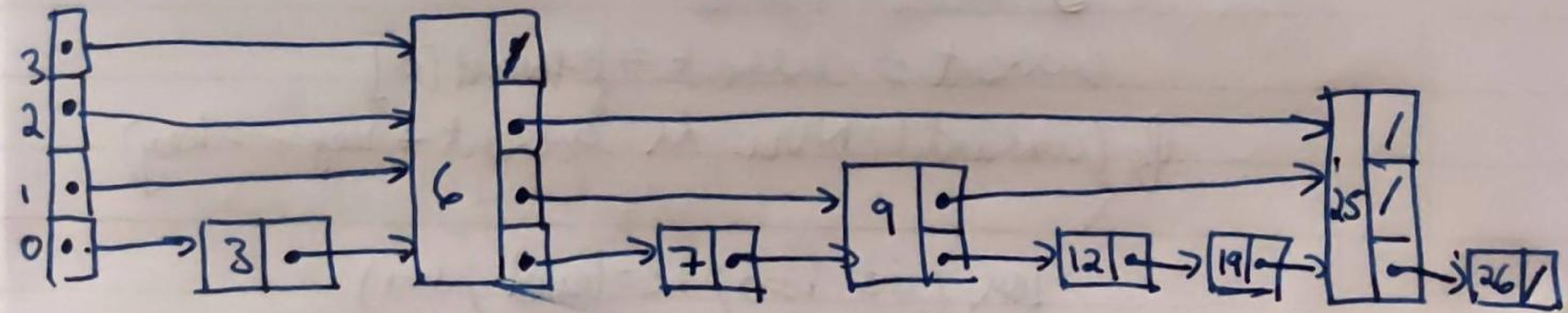
Scanned by TapScanner

```
Node * SkipList ::createNode (int key, int level)
  {
      Node *n = new Node (key, level);
      return n;
    };

void SkipList :: insertElement (int key)
{
     Node  *current = header;
        Node  * update [MAXLVL +1];
        memset (update, 0 , sizeof (Node *)*
                                      (MAXLVL+1

    for(int i = level; i>=0; i--)
      {
          while (current→forward [i] != NULL &&
               current→forward[i]→key < key)
               current= current→forward [i];
           update [i] = current;
        }
  current= current→forward[0];
   ~~if  slevel = randomLevel();~~

    if if (current== NULL || current→key != key)
          {
            int slevel = randomLevel();
if (rlevel > level)
  {
    for (int i = level +1; i < rlevel +1; i++)
              update[i] = header;

      level = rlevel;
  }
 Node * n = createNode (key, rlevel)

  for (int i=0; i <= rlevel; i++)
      {
        n->forward[i] = update[i] ->forward[i];
            update[i]-> forward[i] = n;
      }
      cout<<"Successfully Inserted key"<< key <<endl;
      }
  };
void SkipList :: displayList()
 {
    for(int i=0; i<=level; i++)
     {
       Node * node = header ->forward[i];
        cout<<"Level "<< i <<": ";
          while(node != NULL)
          {
             cout<< node->key<<" ";
              node = node -> forward[i];
          }
  }; }  cout<<endl;
```

```
void SkipList :: deleteElement (int key)
 {
   Node * current = header
     Node * update [MAXLVL + 1];
   memset (update, 0, sizeof (Node *) * (MAXLVL + 1));

   for (int i = level ;  i >= 0; i --)
     {
         while (current -> forward [i] != NULL &&
            current -> forward [i] -> key < key)
            current = current -> forward [i];
          update [i] = current;
       }
       current = current -> forward [0];
     if (current != NULL && current -> key == key)
       {
          for (int i = 0; i <= level ; i++)
            {
               if (update [i] -> forward [i] != current)
                     break;
               update [i] -> forward [i] = current -> forward [i]
            }
         while (level > 0 && header -> forward [level] == 0)
               level --;
            cout << " Successfully deleted key " << key << endl.
     }
 };
```

```
void Skiplist:: searchElement (int key)
 {
   node *current = header;
    for (int i = level ; i>=0; i--)
     {
        while (current→ forward[i] &&
                 current→forward [i]→key < key)
             current= current →forward[i];
        }
       current= current→forward [0];
     if (current and current → key == key)
           cout<< "Found key:" <<key<<
                                     endl;
        };
```

Original Skip list



Level 3:- 3 6

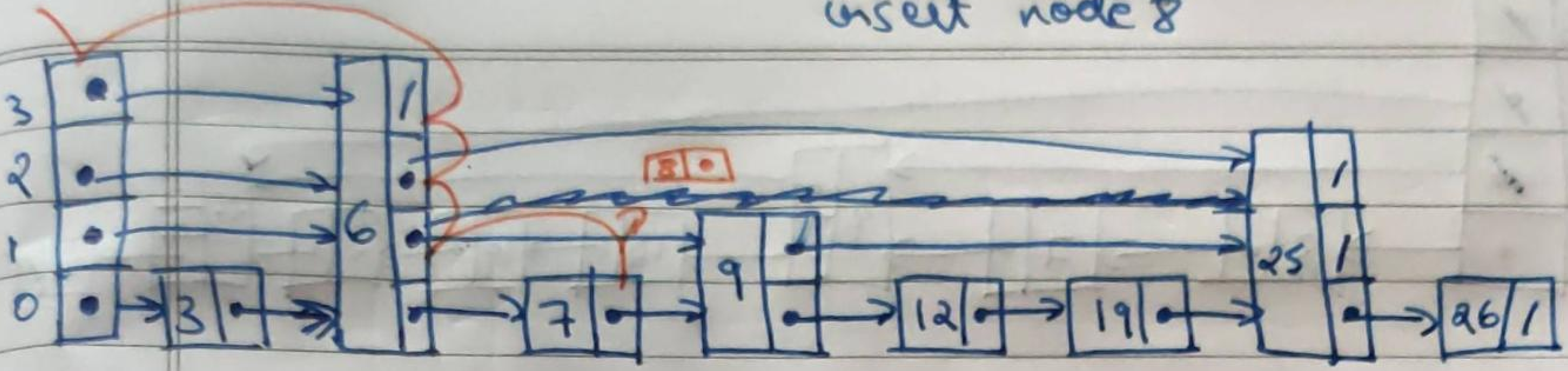
Level2:- 2 6 25

Level 1:- 1 6 9 25

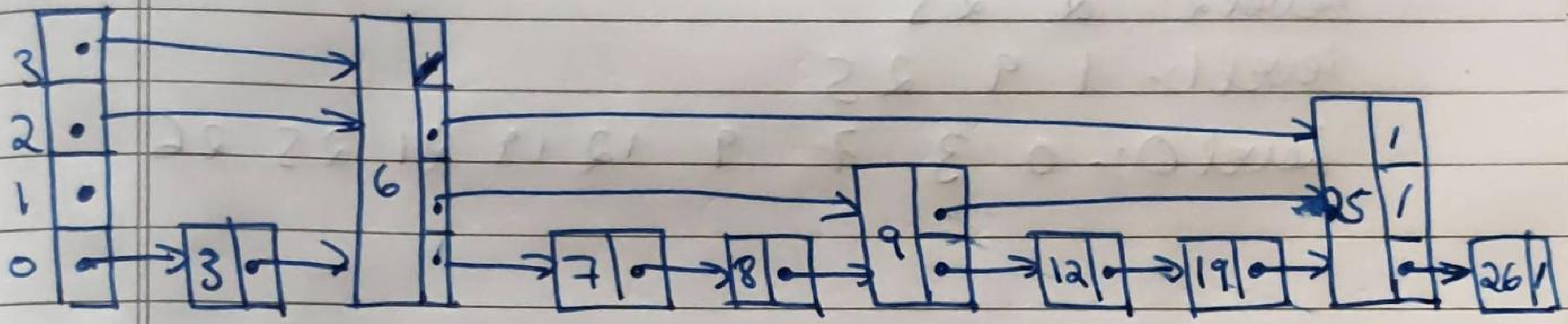
Level 0:- 0 3 6 7 9 12 19 21 25 26

Inserting a node into skip list

insert node 8

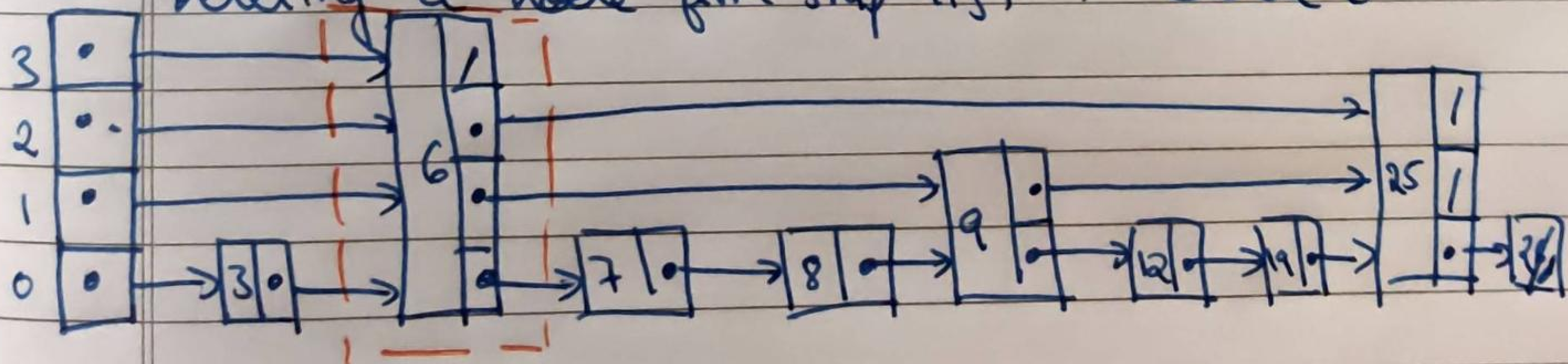


After inserting it will look like

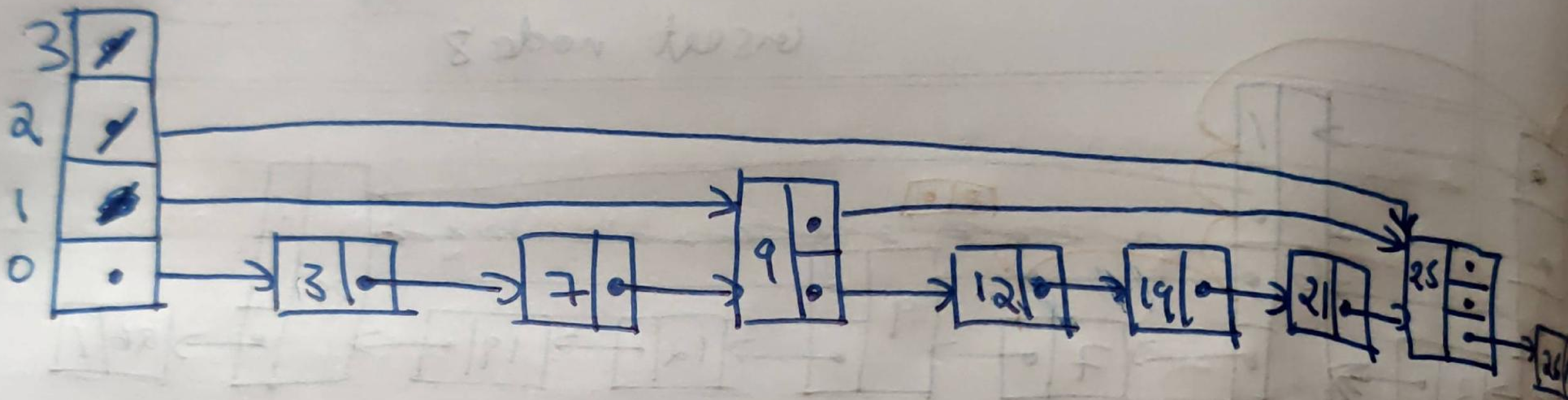


Level 3:- 3 6
Level 2:- 2 6 25
Level 1:- 1 6 9 25
Level 0:- 0 3 6 7 8 9 12 19 21 25 26

Deleting a node from skip list :- delete 6



after deletion



Level 3: 3
Level 2: 2 25
Level 1:- 1 9 25
Level 0:- 0 3 7 9 12 19 21 25 26