Week -6

AIM: A program to implement skip list and its operations (insertion, Display)

DESCRIPTION:

A skip list is a probabilistic data structure. The skip list is used to store a sorted list of elements or data with a linked list. It allows the process of the elements or data to view efficiently. In one single step, it skips several elements of the entire list, which is why it is known as a skip list. It is an extended version of the linkedlist. It allows the user to search, remove and insert the element very quickly. It consists of different levels where the elements are arranged randomly which makes the search operations easier.

Insertion operation:

Insert(list, searchKey)

```
1.  local update[0...MaxLevel+1]
2.  x := list -> header
3.  for i := list -> level downto 0 do
4.  while x -> forward[i] -> key  forward[i]
5.  update[i] := x
6.  x := x -> forward[0]
7.  lvl := randomLevel()
8.  if lvl > list -> level then
9.  for i := list -> level + 1 to lvl do
10. update[i] := list -> header
11. list -> level := lvl
12. x := makeNode(lvl, searchKey, value)
13. for i := 0 to level do
14. x -> forward[i] := update[i] -> forward[i]
15. update[i] -> forward[i] := x
```

Display operation:

```
1.  def display(self):
```

```
2.  ptr=self.header
3.  for l in range(self.level+1):
4.  print("Elements  at level",l,":",end=" ")
5.  node=ptr.next[l]
6.  while(node!=None):
a.  print(node.data,end=" ")
b.  node=node.next[l]
7.  print("\n")
```

PROGRAM:

```
from random import *
class Node:
    def __init__(self,data,level):
        self.data=data
        self.next=[None]*(level+1)
class Skiplist:
    def __init__(self,maxlevel,P):
        self.maxlevel=maxlevel
        self.P=P
        self.header=Node(-1,maxlevel)
        self.level=0
    def randomlevel(self):
        lvl=0
        while random()<self.P and lvl<self.maxlevel:
            lvl+=1
        return lvl
    '''def randomlevel(self):
        lvl=randint(0,self.maxlevel)
```

```
        return lvl'''
    def insertion(self):
        key=int(input("Enter value to be inserted : "))
        update=[None]*(self.maxlevel+1)
        curr=self.header
        for x in range(self.level,-1,-1):
            while curr.next[x] and curr.next[x].data<key:
                curr=curr.next[x]
            update[x]=curr
        curr=curr.next[0]
        if curr==None or curr.data!=key:
            rlevel=self.randomlevel()
            if rlevel>self.level:
                for i in range(self.level+1,rlevel+1):
                    update[i]=self.header
                self.level=rlevel
        n=Node(key,rlevel)
        for x in range(rlevel+1):
            n.next[x]=update[x].next[x]
            update[x].next[x]=n
        print("inserted ",key)
    def display(self):
        ptr=self.header
        for l in range(self.level+1):
            print("Elements  at level",l,":",end=" ")
            node=ptr.next[l]
            while(node!=None):
```

```
            print(node.data,end=" ")
            node=node.next[l]
        print("\n")
n=int(input("Enter no of values to be inserted: "))
maxlevel=2**(n-1)
list=Skiplist(maxlevel,0.4)
for x in range(n):
    list.insertion()
list.display()
```

OUTPUT:

```
rograms\week6_1_skiplist.py
Enter no of values to be inserted: 5
Enter value to be inserted : 1
inserted  1
Enter value to be inserted : 2
inserted  2
Enter value to be inserted : 3
inserted  3
Enter value to be inserted : 4
inserted  4
Enter value to be inserted : 5
inserted  5
Elements  at level 0 : 1 2 3 4 5

Elements  at level 1 : 3 5

Elements  at level 2 : 3
```

TIME COMPLEXITY:

Insertion operation:

Average case: O(log n)

Worst case: O(n)

Display operation:

Average and worst case: O(n)

CONCLUSION: The code is error free and runs as expected.