ZEAL EDUCATION SOCIETY'S

ZEAL COLLEGE OF ENGINEERING
AND RESEARCH

NARHE |PUNE -41 | INDIA

Department of Computer Engineering

# Data Structures and Algorithms Laboratory

ASSIGNMENT NO: 2

**NAME: PAWAR SUJEET UDDHAV**

**CLASS: SE-C**

**ROLL NO: S213009**

## Group A: Assignment No:03

**Problem Statement:**

**For given set of elements create skip list. Find the element in the set that is closest to some given value. (note: Decide the level of element in the list Randomly with some upper limit)**

**Program:**

```
import random
class
Node(object):
   def__init__(self,key,level):
      self.key= key
      self.forward=[None]*(level+1
      )
class Skiplist(object):
   def__init
      ___(self,max_lvl,P):
      self.MaxLvl= max_lvl
      self.P=P
      self.header=self.createNode(self.MaxLvl,-
      1) self.level=0
   #create new node
   def
      createNode(self,lvl,key):
      n=Node(key,lvl)
      return n
   # create random level for
   node def random_event(self):
      lvl=0
```

```
    while random.random()<self.P and
        lvl<self.MaxLvl: lvl+=1
    return lvl

# insert given key in skip
list def
insertElement(self,key):
    update=
    [None]*(self.MaxLvl+1) current
    = self.header
    for i in range(self.level,-1,-1):
        while current.forward[i] and current.forward[i].key
            <key: current=current.forward[i]
        update[i]=current
    current=
    current.forward[0]
    if current == None or
        current.key!=key: rlevel =
        self.random_event()
    if rlevel>self.level:
        for i in
            range(self.level+1,rlevel+1):
            update[i]=self.header
        self.level=rlevel
    n= self.createNode(rlevel,key)
    # insert node by rearranging
    ref for i in range(rlevel+1):
        n.forward[i]=
        update[i].forward[i]
        update[i].forward[i]=n
```

```
   print("Successfully Inserted Key
{}".format(key)) def
deleteElement(self,search_key):
   update=[None]*(self.MaxLvl+1)
# insert given key in skip
list def
insertElement(self,key):
   update=
   [None]*(self.MaxLvl+1) current
   = self.header
   for i in range(self.level,-1,-1):
      while current.forward[i] and current.forward[i].key
         <key: current=current.forward[i]
      update[i]=current
   current=
   current.forward[0]
   if current == None or
      current.key!=key: rlevel =
      self.random_event()
   if rlevel>self.level:
      for i in
         range(self.level+1,rlevel+1):
         update[i]=self.header
      self.level=rlevel
   n= self.createNode(rlevel,key)
   # insert node by rearranging
   ref for i in range(rlevel+1):
      n.forward[i]=
      update[i].forward[i]
      update[i].forward[i]=n
```

```
    print("Successfully Inserted Key
{}".format(key)) def
deleteElement(self,search_key):
    update=[None]*(self.MaxLvl+1)
    current= self.header
    for i in range(self.level,-1,-1):
        while (current.forward[i]and
            current.forward[i].key<search_key):
            current=current.forward[i]
        update[i]= current

# insert given key in skip
list def
insertElement(self,key):
    update=
    [None]*(self.MaxLvl+1) current
    = self.header
    for i in range(self.level,-1,-1):
        while current.forward[i] and current.forward[i].key
            <key: current=current.forward[i]
        update[i]=current
    current=
    current.forward[0]
    if current == None or
        current.key!=key: rlevel =
        self.random_event()
    if rlevel>self.level:
        for i in
            range(self.level+1,rlevel+1):
            update[i]=self.header
```

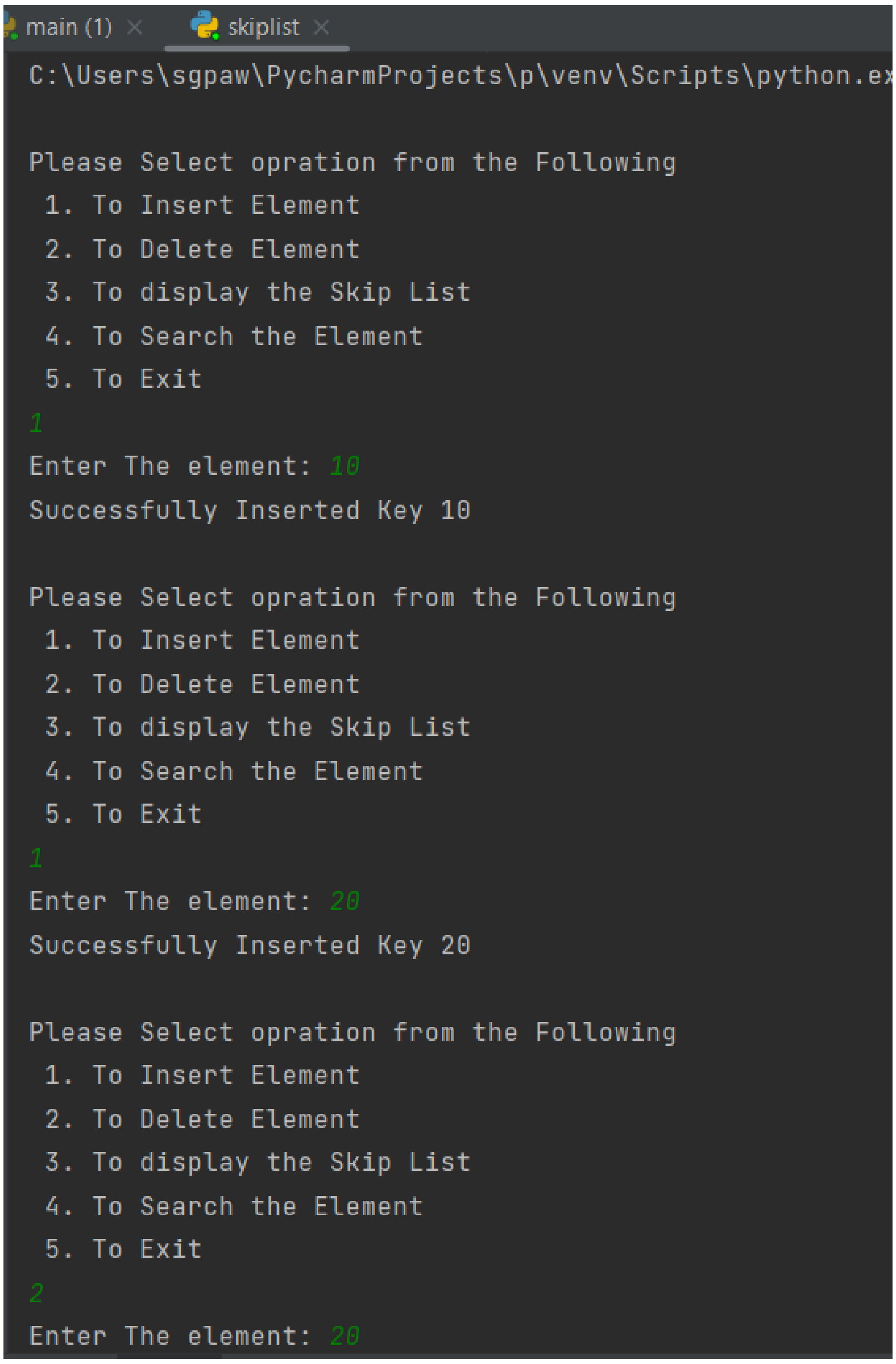
```
          self.level=rlevel

      n= self.createNode(rlevel,key)
      # insert node by rearranging
      ref for i in range(rlevel+1):
          n.forward[i]=
          update[i].forward[i]
          update[i].forward[i]=n
      print("Successfully Inserted Key
  {}".format(key)) def
  deleteElement(self,search_key):
      update=[None]*(self.MaxLvl+1)
      current= self.header
      for i in range(self.level,-1,-1):
          while (current.forward[i]and
              current.forward[i].key<search_key):
              current=current.forward[i]
          update[i]= current

      current=current.forward[0]
      # if current node is targeted node
      if current!= None and current.key ==
          search_key: for i in range(self.level+1):
              if
                  update[i].forward[i]!=current
                  : break
              update[i].forward[i]=current.forward[i
          ] # remove level having no elements
          while (self.level>0 and
              self.header.forward[self.level]==None): self.level-=1
          print("Successfully delete {}
".format(search_key)) def searchElement(self,key):
    current = self.header
    for i in range(self.level,-1,-1):
        while (current.forward[i] and
            current.forward[i].key<key): current=
            current.forward[i]
    current = current.forward[0]
    # if current node have key equal to search
    key if current and current.key ==key:
        print("Found key ",
key) # display skip list
def displayList(self):
    print("\n*******Skip
    List********") head= self.header
    for lvl in range(self.level+1):
        print("Level {}: ".format(lvl),end="
        ") node = head.forward[lvl]
        while (node != None):

            print(node.key,end=" ")
            node =
            node.forward[lvl]
        print(""
)   def main():
lst =
Skiplist(3,0.5)
start= 0
while start!=5:
    opr= input("\nPlease Select opration from the Following\n 1. To Insert
Element \n 2. To Delete Element \n 3. To display the Skip List \n 4. To Search
the Element\n 5. To Exit \n")
```

```
        if opr=="1":
            element = int(input("Enter The element:
            ")) lst.insertElement(element)
        elif opr =="2":
            element = int(input("Enter The element:
            ")) lst.deleteElement(element)
        elif opr=="3":
            lst.displayList(
            )
        elif opr=="4":
            element = int(input("Enter The element:
            ")) lst.searchElement(element)
        elif
            opr=="4":
            start=5
        else:
            print("Select Correct Operation!")
main()
```

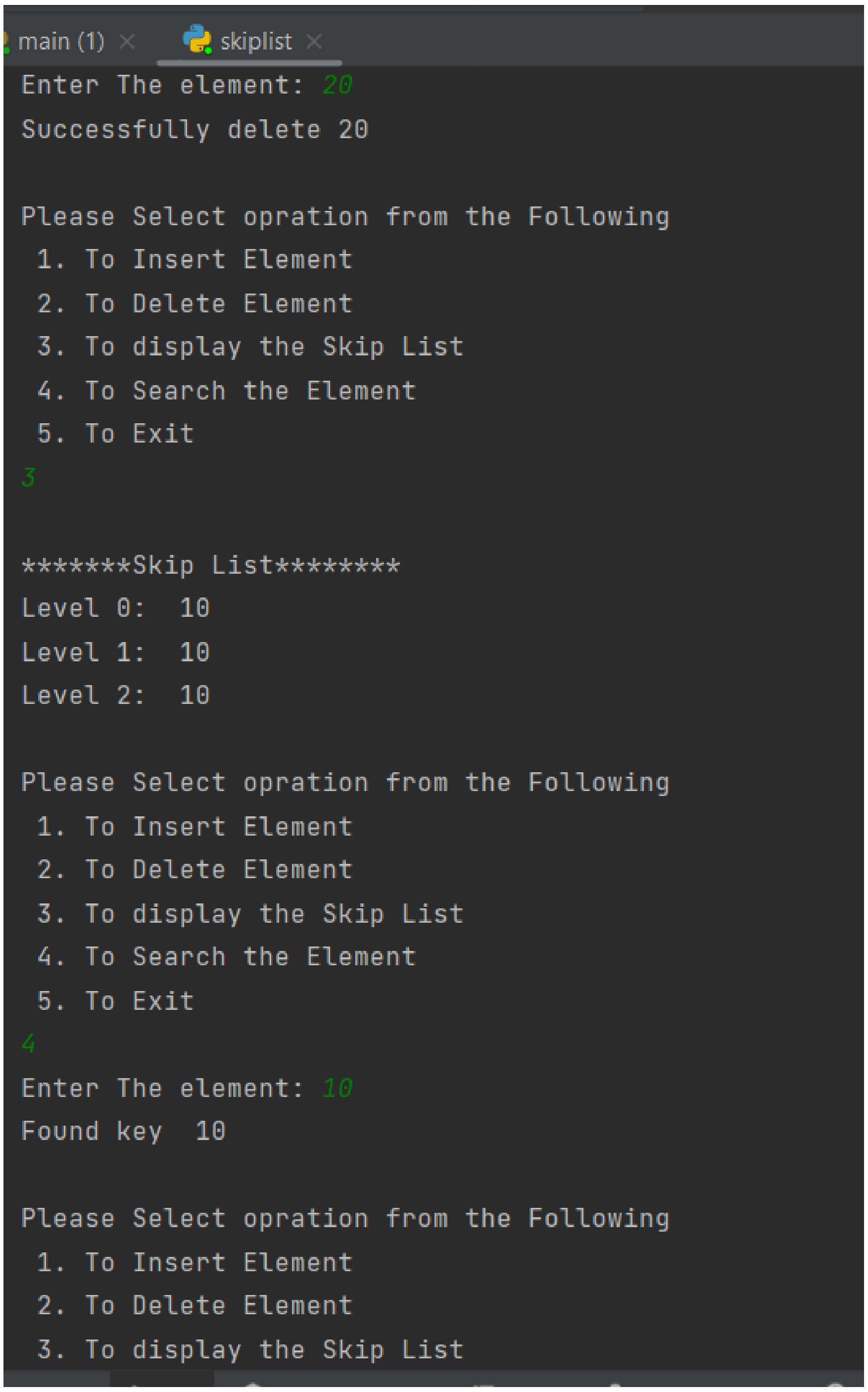


```
C:\Users\sgpaw\PycharmProjects\p\venv\Scripts\python.ex

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
1
Enter The element: 10
Successfully Inserted Key 10

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
1
Enter The element: 20
Successfully Inserted Key 20

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
2
Enter The element: 20
```



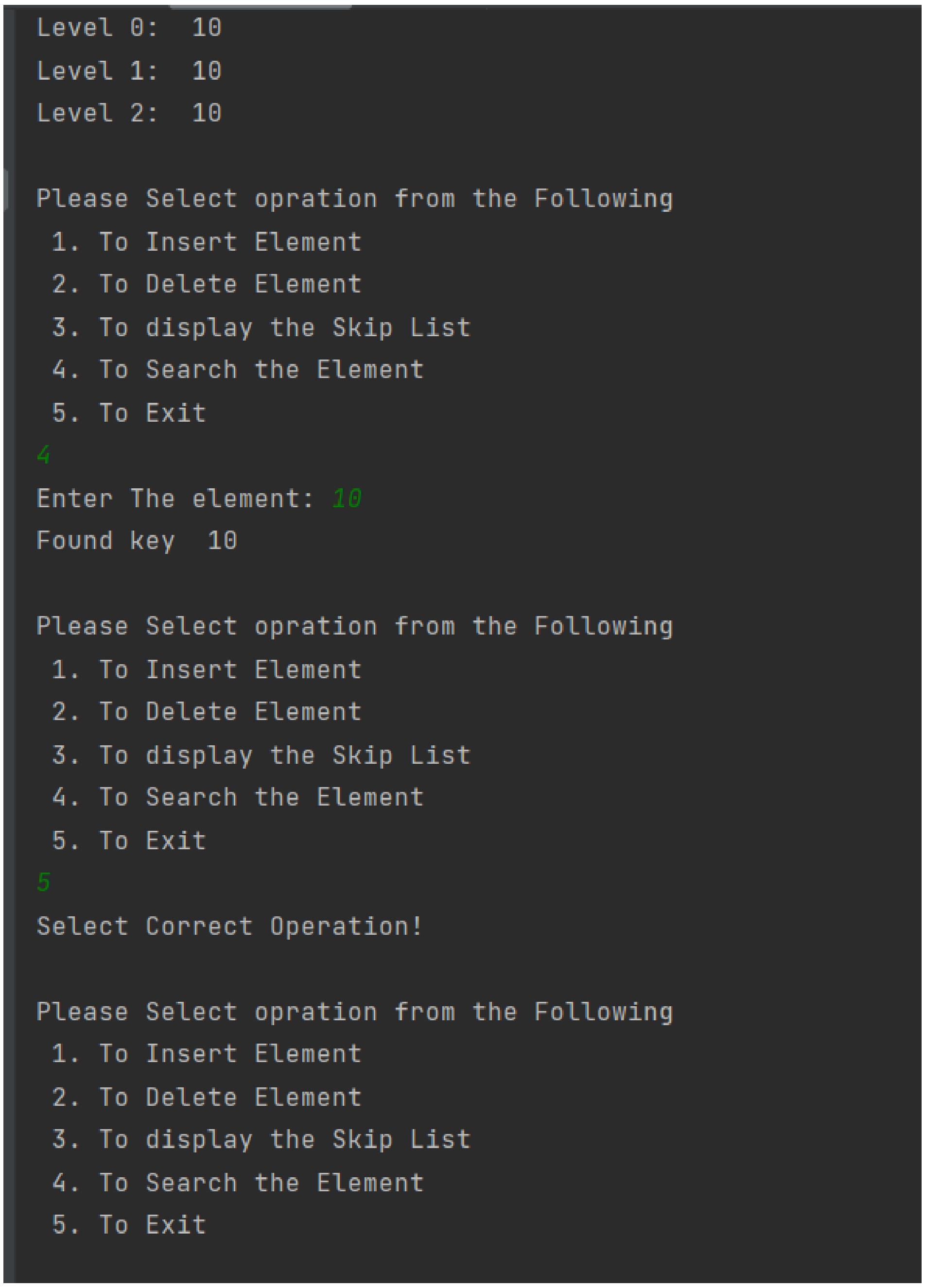
```
Enter The element: 20
Successfully delete 20

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
3


*******Skip List*********
Level 0:  10
Level 1:  10
Level 2:  10

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
4
Enter The element: 10
Found key  10

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
```



```
Level 0:  10
Level 1:  10
Level 2:  10

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
4
Enter The element: 10
Found key  10

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
5
Select Correct Operation!

Please Select opration from the Following
 1. To Insert Element
 2. To Delete Element
 3. To display the Skip List
 4. To Search the Element
 5. To Exit
```