In [1]:

```
#lab : 5
#Pgm : 1
#case : 1
print("case : 1")
print()
def find_averge(s):
    print("Roll No. : ",s[0])
    print("Name : ",s[1])
    print("Mark1 : ",s[2])
    print("Mark2 : ",s[3])
    print("Mark3 : ",s[4])
    for i in s:
        m=s[2]+s[3]+s[4]
    print("Average Mark :",m)

#main:
stud=(1,"Rex",60,85,70)
stud1=tuple(stud)
find_averge(stud1)
```

```
case : 1

Roll No. :  1
Name :  Rex
Mark1 :  60
Mark2 :  85
Mark3 :  70
Average Mark : 215
```

In [4]:
```
#lab : 5
#Pgm : 1
#case : 2
print("case : 2")
print()

def find_averge(s):
    print("Roll No. : ",s[0])
    print("Name : ",s[1])
    print("Mark1 : ",s[2][0])
    print("Mark2 : ",s[2][1])
    print("Mark3 : ",s[2][2])
    for i in s:
        m=s[2][0]+s[2][1]+s[2][2]
    print("Average Mark :",m)
#main:
stud=(2,"Rex",(80,75,90))
stud2=tuple(stud)
find_averge(stud2)
```

```
case : 2

Roll No. :  2
Name :  Rex
Mark1 :  80
Mark2 :  75
Mark3 :  90
Average Mark : 245
```

In [ ]:

In [10]:

```
n=int(input("No. of Values : "))
a=[int(input("Values : "))for i in range(n)]
b=[]
num=int(input('No. of Bigest Number : '))

for x in range(0,num):
    N=0
    for y in range(len(a)):
        if a[y]>N:
            N=a[y]
    a.remove(N)
    b.append(N)
print(num,"Bigest No.:")
for i in b:
    print(i)
```

```
No. of Values : 5
Values : 1
Values : 2
Values : 33
Values : 12
Values : 10
No. of Bigest Number : 3
3 Bigest No.:
33
12
10
```

In [ ]:

In [4]:
```
#Pgm:5
def Sort_last(t):
    l=len(t)
    for i in range(0,l):
         for j in range(0,l-i-1):
            if (t[j][1] > t[j + 1][1]):
                temp = t[j]
                t[j]= t[j + 1]
                t[j + 1]=temp
    return t

#Main:
tp=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,20)]
print("Input :",tp)
print("Ouput :")
print(Sort_last(tp))
```

```
Input : [(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 20)]
Ouput :
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 20), (10, 7, 15)]
```

In [10]:
```
#pgm:4

def Sort(lst):
    l=lst
    x=[]
    nox=[]
    for i in l:
        if i[0].lower() == "x" :
            x.append(i)
        else:
            nox.append(i)
    x.sort(),nox.sort()
    return x + nox

#main:
lst1=['mix', 'xyz', 'apple', 'xanadu', 'aardvark']
lst2=['ccc','bbb','aaa','xcc','xaa']
lst3=['bbb', 'ccc', 'axx', 'xzz', 'xaa']
print("Input :",lst1)
print("Output :",Sort(lst1))
print("Input :",lst3)
print("Output :",Sort(lst3))
```

```
Input : ['mix', 'xyz', 'apple', 'xanadu', 'aardvark']
Output : ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
Input : ['bbb', 'ccc', 'axx', 'xzz', 'xaa']
Output : ['xaa', 'xzz', 'axx', 'bbb', 'ccc']
```

In [ ]:

In [6]:
```
#pgm :6(a)

def first(s):
    print(s[0])
#main:
t=(100,2,3,4,5,6,7,8,9,0)
print("The first element of the Tuble : ")
first(t)
```

```
The first element of the Tuble :
100
```

In [7]:
```
#pgm :6(b)
def sort_first(s):
    return sorted(s)
#main:
t=[(4,1,5),(9,4,3),(1,2,3),(10,23,5)]
print("Sorted List :")
print(sort_first(t))
```

```
[(1, 2, 3), (4, 1, 5), (9, 4, 3), (10, 23, 5)]
```

In [9]:
```
#pgm :6(c)
def sort_first(s):
    print("Sorted List :",sorted(s))
#main:
t=[(4,1,5),(9,3),(1,2),(10,23,5)]
sort_first(t)
```

```
Sorted List : [(1, 2), (4, 1, 5), (9, 3), (10, 23, 5)]
```

In [17]:
```
#pgm :6(d)
def middle(s):
    a=len(s)/2
    print(s[int(a)])
#main:
t=(100,21,34,4,500)
print("Middle element of Tuble :")
middle(t)
```

```
Middle element of Tuble :
34
```

In [ ]:

In [13]:
```
#lab : 5
#Pgm : 2

a=[]
d1=float(input("Day 1 : "))
d2=float(input("Day 2 : "))
d3=float(input("Day 3 : "))
d4=float(input("Day 4 : "))
d5=float(input("Day 5 : "))
d6=float(input("Day 6 : "))
d7=float(input("Day 7 : "))
a.append(d1)
a.append(d2)
a.append(d3)
a.append(d4)
a.append(d5)
a.append(d6)
a.append(d7)
print("First Day Weight : ",a[0])
print("First Day Weight : ",a[6])
print("Highest Weight : ",max(a))
print("Lowest Weight : ",min(a))
print("Averge Weight : ",sum(a)/len(a))
if sum(a)/len(a)<min(a):
    print("Your Weight Management is Excellent")
else:
    print("Your Weight Management is NOT So Good. Please take Care of your DIET"
```

```
Day 1 : 110
Day 2 : 108
Day 3 : 105
Day 4 : 102
Day 5 : 100
Day 6 : 98
Day 7 : 95
First Day Weight :  110.0
First Day Weight :  95.0
Highest Weight :  110.0
Lowest Weight :  95.0
Averge Weight :  102.57142857142857
Your Weight Management is NOT So Good. Please take Care of your DIET
```

In [14]:
```
#lab : 5
#pgm :6.f
def Sort_last(t):
    l=len(t)
    for i in range(0,l):
         for j in range(0,l-i-1):
            if (t[j][1] > t[j + 1][1]):
                temp = t[j]
                t[j]= t[j + 1]
                t[j + 1]=temp
    return t

#Main:
tp=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,20)]
print("Input :",tp)
print("Ouput :")
print(Sort_last(tp))
```

```
Input : [(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 20)]
Ouput :
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 20), (10, 7, 15)]
```

In [11]:
```
#lab : 5
#pgm:7

def remove_adjacent(l):
    li=list(dict.fromkeys(l))
    print("Output :")
    print(li)
#main:
l1=[1,2,2,3]
l2=[2,2,3,3,3]
l3=[]
l4=[2,5,5,6,6,7]
l5=[6,7,7,8,9,9]
print("Input :",l1)
remove_adjacent(l1)
print("Input :",l2)
remove_adjacent(l2)
print("Input :",l3)
remove_adjacent(l3)
print("Input :",l4)
remove_adjacent(l4)
print("Input :",l5)
remove_adjacent(l5)
```

```
Input : [1, 2, 2, 3]
Output :
[1, 2, 3]
Input : [2, 2, 3, 3, 3]
Output :
[2, 3]
Input : []
Output :
[]
Input : [2, 5, 5, 6, 6, 7]
Output :
[2, 5, 6, 7]
Input : [6, 7, 7, 8, 9, 9]
Output :
[6, 7, 8, 9]
```

In [21]:
```
#lab : 5
#pgm : 8

def verbing(w):
    l=len(w)
    if l<=2:
        print(w)
    if l >= 3:
        if w[-3:] == 'ing':
            w+='ly'
    else:
        w+='ing'
    print(w)
#main:
verbing('hail')
verbing('swimming')
verbing('do')
```

```
hail
swimmingly
do
doing
```

In [ ]:

In [6]:

```
#Lab :5
#Pgm :9

def not_bad(str1):
    nt=str1.find('not')
    bd = str1.find('bad')
    if bd>nt and nt>0 and bd>0:
        str1 = str1.replace(str1[nt:(bd+4)], 'good')
        return str1
    else:
        return str1
#main:
str2='This dinner is not that bad'
print("Input :",str2)
print("Ouput :",not_bad(str2))
```

```
Input : This dinner is not that bad
Ouput : This dinner is good
```

In [ ]: