Question3. Write a function lastN(lst, n) that takes a list of integers and n and returns n largest numbers.

How many numbers you want to enter?: 6 Enter a number: 12 Enter a number: 32 Enter a number: 10 Enter a number: 9 Enter a number: 52 Enter a number: 45 How many largest numbers you want to find?: 3 Largest numbers are: 52, 45, 32

In [1]:

```
def lastN(lst,n):
    final_list=[]

    for i in range(n):
        max1 =0

        for j in lst:
            if j>max1:
                max1 =j

        lst.remove(max1);
        final_list.append(max1)
    print("\nlargest number:",final_list)
b=int(input("How many number want to enter?:"))
lst=[]
for i in range(b):
    c=int(input('\nEnter a number: '))
    lst.append(c)

n=int(input("\nHow many largest number you want to find?:"))
lastN(lst,n)
```

```
How many number want to enter?:6

Enter a number: 12

Enter a number: 32

Enter a number: 10

Enter a number: 9

Enter a number: 52

Enter a number: 45

How many largest number you want to find?:3

largest number: [52, 45, 32]
```

Question4. Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with 'x' first. Hint: this can be done by making 2 lists and sorting each of them before combining them.

Test Cases:

1. **Input: ['mix', 'xyz', 'apple', 'xanadu', 'aardvark'] Output: ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']**
2. **Input: [‘ccc’,’bbb’,’aaa’,’xcc’,’xaa’] Output: [‘xaa’,’xcc’,’aaa’,’bbb’,’ccc’]**
3. **Input: [‘bbb’,’ccc’,’axx’,’xzz’,’xaa’] Output: [‘xaa’,’xzz’,’axx’,’bbb’,’ccc’]**

In [3]:

```
def front_x(words):
    xlist=[]
    alist=[]

    for word  in words:
        if word.startswith("x"):
            xlist.append(word)
```

```
        else:
            alist.append(word)
    return sorted(xlist)+ sorted(alist)
```

In [4]:

```
words=['min','xyz','apple','xanadu','aardvark']
front_x(words)
```

Out[4]:

```
['xanadu', 'xyz', 'aardvark', 'apple', 'min']
```

In [5]:

```
words=['ccc','bbb','aaa','xcc','xaa']
front_x(words)
```

Out[5]:

```
['xaa', 'xcc', 'aaa', 'bbb', 'ccc']
```

In [6]:

```
words=['bbb','ccc','axx','xzz','xaa']
front_x(words)
```

Out[6]:

```
['xaa', 'xzz', 'axx', 'bbb', 'ccc']
```

Question5. Develop a function sort_last(). Given a list of non-empty tuples, return a list sorted in increasing order by the last element in each tuple. Hint: use a custom key= function to extract the last element form each tuple.

Test Cases:

1. **Input: [(1, 7), (1, 3), (3, 4, 5), (2, 2)] Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]**
2. **Input: [(1,3),(3,2),(2,1)] Output: [(2,1),(3,2),(1,3)]**
3. **Input: [(2,3),(1,2),(3,1)] Output: [(3,1),(1,2),(2,3)]**

In [7]:

```
def last(n): return n[-1]

def sort_list_last(tuples):
    return sorted(tuples, key=last)

print("Test Cases:1 \n\tInput:[(1,7),(1,3),(3,4,5),(2,2)] \n\tOutput:",sort_list_last([(1
,7),(1,3),(3,4,5),(2,2)]))

print("Test Cases:2 \n\tInput:[(1,3),(3,2),(2,1)] \n\tOutput:",sort_list_last([(1,3),(3,2
),(2,1)]))

print("Test Cases:3 \n\tInput:[(2,3),(1,2),(3,1)] \n\tOutput:",sort_list_last([(2,3),(1,2
),(3,1)]))
```

```
Test Cases:1
 Input:[(1,7),(1,3),(3,4,5),(2,2)]
 Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
Test Cases:2
 Input:[(1,3),(3,2),(2,1)]
 Output: [(2, 1), (3, 2), (1, 3)]
Test Cases:3
 Input:[(2,3),(1,2),(3,1)]
 Output: [(3, 1), (1, 2), (2, 3)]
```

Question6. Other String Functions a) Define a function first() that receives a tuple and returns its first element b) Define a function sort_first() that receives a list of tuples and returns the sorted c) Print lists in sorted order d) Define a function middle() that receives a a tuple and returns its middle element e) Define a functino sort_middle() that receives a list of tuples and returns it sorted using the key middle f) Print the list [(1,2,3),

(2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order. Output should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]

In [54]:

```
def first(num):
    return(num[0])

def sort_first(num):
    new_list=sorted(num)
    return new_list

def lists(num):
    return sorted(num, key=None, reverse=0)


def middle(num):
    return(num[int(len(num)/2)])

def sorte_middle(num):
    return sorted(num,key=middle)
```

In [55]:

```
num=[(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)]
first(num)
```

Out[55]:

```
(1, 2, 3)
```

In [56]:

```
sort_first(num)
```

Out[56]:

```
[(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 40)]
```

In [57]:

```
lists(num)
```

Out[57]:

```
[(1, 2, 3), (2, 1, 4), (10, 7, 15), (20, 4, 50), (30, 6, 40)]
```

In [58]:

```
middle(num)
```

Out[58]:

```
(10, 7, 15)
```

In [59]:

```
sorte_middle(num)
```

Out[59]:

```
[(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]
```

Question7. Develop a function remove_adjacent(). Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

1. **Input: [1, 2, 2, 3] and output: [1, 2, 3]**
2. **Input: [2, 2, 3, 3, 3] and output: [2, 3]**
3. **Input: [ ]. Output: [ ]. 4. Input: [2,5,5,6,6,7] Output: [2,5,6,7]**

4. Input: [6,7,7,8,9,9] Output: [6,7,8,9]

4. Input: [6,7,7,8,9,9] Output: [6,7,8,9]

In [9]:

```
def remove_adjacent(nums):

    result =[]
    for num in nums:
        if len(result)==0 or num != result[-1]:
            result.append(num)
    return result
```

In [10]:

```
nums=[1,2,2,3]
remove_adjacent(nums)
```

Out[10]:

```
[1, 2, 3]
```

In [11]:

```
nums=[2,2,3,3]
remove_adjacent(nums)
```

Out[11]:

```
[2, 3]
```

In [12]:

```
nums=[]
remove_adjacent(nums)
```

Out[12]:

```
[]
```

In [13]:

```
nums=[6,7,7,8,9,9]
remove_adjacent(nums)
```

Out[13]:

```
[6, 7, 8, 9]
```

In [ ]:

Question3. Write a function **lastN(lst, n)** that takes a list of integers and **n** and returns **n** largest numbers.

```
How many numbers you want to enter?: 6
Enter a number: 12
Enter a number: 32
Enter a number: 10
Enter a number: 9
Enter a number: 52
Enter a number: 45
How many largest numbers you want to find?: 3
Largest numbers are: 52, 45, 32
```

```
def lastN(lst,n):
    final_list=[]
    for i in range(n):
        max1=0
        for j in lst:
            if j>max1:
                max1=j
        lst.remove(max1)
        final_list.append(max1)
    print("\nlargest number:",final_list)

b=int(input("How many numbers want to enter?:"))
lst=[]
for i in range(b):
    c=int(input("\n Enter a number:"))
    lst.append(c)
n=int(input("\n How many largest number you want to find?:")) lastN(lst,n)
```

Question4. Given a list of strings, return a list with the strings in sorted order, except group all the strings that begin with 'x' first. Hint: this can be done by making 2 lists and sorting each of them before combining them.

Test Cases:

1. Input: ['mix', 'xyz', 'apple', 'xanadu', 'aardvark']
   Output: ['xanadu', 'xyz', 'aardvark', 'apple', 'mix']
2. Input: ['ccc','bbb','aaa','xcc','xaa']
   Output: ['xaa','xcc','aaa','bbb','ccc']
3. Input: ['bbb','ccc','axx','xzz','xaa']
   Output: ['xaa','xzz','axx','bbb','ccc']

```
def front_x(words):
    x list=[]
    a list=[]
    for word in words:
        if word.startswith("x"):
            x list.append(word)
        else:
            a list.append(word)
    return sorted(x list)+sorted(a list)
words=['mix','xyz','apple','xanadu','aardvark']
front_x(words)
words=['ccc','bbb','aaa','xcc','xaa']
front_x(words)
```

```
words=['bbb','ccc','axx','xzz','xaa']
front_x(words)
```

Question5. Develop a function **sort_last()**. Given a list of non-empty tuples, return a list sorted in increasing order by the last element in each tuple. Hint: use a custom key - function to extract the last element form each tuple.

Test Cases:

1. Input: [(1, 7), (1, 3), (3, 4, 5), (2. 2)]
   Output: [(2, 2), (1, 3), (3, 4, 5), (1, 7)]
2. Input: [(1,3),(3,2),(2,1)]
   Output: [(2,1),(3,2),(1,3)]
3. Input: [(2,3),(1,2),(3,1)]
   Output: [(3,1),(1,2),(2,3)]

```
print("Test Cases:\n\t Input: [(2,3),(1,2),(3,1)]
\n\t Output:", sort_list_last([(2,3),(1,2),(3,1)]))

def last(n): return n[-1]

def sort_list_last(tuples):
    return sorted(tuples, key=last)

print("Test cases:1 \n\t Input: [(1,7),(1,3),(3,4,5),(2,2)] \n\t output:", sort_list_last([(1,7),(1,3),(3,4,5),(2,2)]))

print("Test cases:2 \n\t Input: [(1,3),(3,2),(2,1)] \n\t Output:", sort_list_last([(1,3),(3,2),(2,1)]))
```

Question6. Other String Functions

a) Define a function **first()** that receives a tuple and returns its first element
b) Define a function **sort_first()** that receives a list of tuples and returns the sorted
c) Print lists in sorted order
d) Define a function **middle()** that receives a a tuple and returns its middle element
e) Define a functino **sort_middle()** that receives a list of tuples and returns it sorted using the key middle
f) Print the list [(1,2,3), (2,1,4), (10,7,15), (20,4,50), (30, 6, 40)] in sorted order. Output should be: [(2, 1, 4), (1, 2, 3), (20, 4, 50), (30, 6, 40), (10, 7, 15)]

```
def first(num):
    return(num[0])

def sort_first(num):
    new_list = sorted(num)
    return new_list

def lists(num):
    return sorted(num, key=None, reverse=0)

def middle(num):
    return(num[int(len(len(num))/2)])

def sorte_middle(num):
    return sorted(num, key=middle)

num=[(1,2,3),(2,1,4),(10,7,15),(20,4,50),(30,6,40)]
first(num)
sort_first(num)

lists(num)
middle(num)
sorte_middle(num)
```

Question7. Develop a function **remove_adjacent()**. Given a list of numbers, return a list where all adjacent same elements have been reduced to a single element. You may create a new list or modify the passed in list.

Test Cases:

1. Input: [1, 2, 2, 3] and output: [1, 2, 3]
2. Input: [2, 2, 3, 3, 3] and output: [2, 3]
3. Input: [ ]. Output: [ ].
4. Input: [2,5,5,6,6,7]
   Output: [2,5,6,7]
5. Input: [6,7,7,8,9,9]
   Output: [6,7,8,9]

```
def remove_adjacent(nums):
    result = []
    for num in nums:
        if len(result) == 0 or num != result[-1]:
            result.append(num)
    return result

nums = [1,2,2,3]
remove_adjacent(nums)

nums: [2,2,3,3]
remove_adjacent(nums)

nums = [2,2,3,3]
remove_adjacent(nums)

nums = [ ]
remove_adjacent(nums)

nums = [6,7,7,8,9,9]
remove_adjacent(nums)
```