In [23]:
```
# Function to print all combinations of all integers in given list
#[1,2,3]->(1,2)(1,3)(2,3) ->3C2 ->3!/(3-2)!* 2!  -->6/2=3

def combinations(li):
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):

                print(li[i],li[j])
combinations([1,2,3,4])

```

```
1 2
1 3
1 4
2 3
2 4
3 4
```

In [22]:
```
def combinations(li):
    for i in range(len(li)-2):
        for j in range(i+1,len(li)-1):
            for k in  range(j+1,len(li)):
                print(li[i],li[j],li[k])
    return
combinations([1,2,3,4,5])
```

```
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

Write a program to k- largest element in given array

In [33]:
```
def klargestdifference(li):
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):
            d=abs(li[i]-li[j])
            if d not in li:
                li.append(d)
    return li
#klargestdifference([1,3])   #o/p:  [1,3,2]
#klargestdifference([1,2])  # o/p:   [1,2]
#klargestdifference([1,8])   # o/p:   [1,8,7]
#klargestdifference([1,5])   # o/p:    [1,5,4]
klargestdifference([1,6])   # o/p:     [1,6,5]
```

Out[33]:
```
[1, 6, 5]
```

Using Cli

In [ ]:

```
def medium(li,k):
    #li3=[[],li]
    #c=1
    while(True):

        li3=klargestdifference(li)
        if li3[0] == li3[1]:
            break

    return sorted(li3[0],reverse=True)[k-1]
    return -1
#return li3[0]
# Function to identify of all pairs of  numbers
# pairs of numbers and add those differences to the same list
# It returns the updted list and original list


def klargestdifference(li):
    cli=li[:]
    newelements=[]
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):
            d=abs(int(li[i])-int(li[j]))
            if d not in li and d not in newelements:
                newelements.append(d)
    li.extend(newelements)
    return [cli,li]
li=[1,9,8,7,6,2]
klargestdifference(li)
#medium(li,2)



with open('DataFiles/medium-input.txt','r') as f:
    t=int(f.readline())
    for i in range(t):
        f.readline()
        li=f.readline().split()
        k=int(f.readline())
        print(medium(li,k))




```

In [13]:
```
# List data
a=[1,2,3]
b=[1,3,2]
a=b.copy()
#a           #Data copy though individual
a=b[:]
a
#a=b

#b.append(4)
#a.append(5)
```

Out[13]:
```
[1, 3, 2]
```

In [11]:
```
#[4,8]
#[20,40,60]
#[4,8,12,16]
#[3,6,9,12]

# Convert the list in to an Arithmetic Progression

def differencepairs(li):
    c=li.copy()
    newlist=[]
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):
            d=abs(li[i]-li[j])
            if d not in li and d not in newlist:
                newlist.append(d)
    li.extend(newlist)
    return [c,li]

li=[2,3,6,9,12,1,4,7,10,5,8,11]   # whenever receive 1 we can stop the list
differencepairs(li)

```

Out[11]:
```
[[2, 3, 6, 9, 12, 1, 4, 7, 10, 5, 8, 11],
 [2, 3, 6, 9, 12, 1, 4, 7, 10, 5, 8, 11]]
```

Set-Data Structure in Python

- is denoted by '{ }'
- it contains a set of values
- it contains only unique elements
- it removes the repeated elements
- it is also immutable
- it does not contain an order

In [35]:
```
a={1,2,3,4,5,6,6}
#a
a.add(7)     # Adding a single element in a set
a
#for i in a:
    #print(i,end=" ")       # Accessing set of elements in a set


b={7,8,9}
li=[11,12,13]
a.update(b,li)   # Adding Multiple elements in a set
a.discard(13)   # Removing the last element in a set
a

```

Out[35]:
```
{8, 9}
```

Set Operations

In [53]:
```
a={10,1,2,3,4,5,6}
b={7,8,9,1,2,3}
a.union(b)
# A U B =B U A
a.intersection(b)
# A ^ B  =B ^ A
c={111,123}
#a.disjoint(b)
#a.isdisjoint(c)

#a-b # All elments of a which are not in
#b-a

sorted(a)

a ^ b   # it represents intersection of either a or b

```

Out[53]:
```
{4, 5, 6, 7, 8, 9, 10}
```

In [55]:
```
d=set()    # Creates an empty set
d
```

Out[55]:
```
set()
```

In [57]:
```
li=[1,2,3,4,2,1,2,3,4,5,6]
u=set(li)
u
```

Out[57]:
```
{1, 2, 3, 4, 5, 6}
```

Functional Programming

- Procedural : C

- Object Oriented : JAVA,Python
- Scripting : PHP,Python,Java Script, Shell, Perl
- Functional : Python,Haskell, Scala
- Logic : Prolog, Lisp

List Comprehensions

In [61]:
```
# List of  N natural numbers
n=10
for i in range(1,n+1):
    print(i,end=" ")
```

```
1 2 3 4 5 6 7 8 9 10
```

In [64]:
```
n=10
l=[]
for i in range(1,n+1):
    l.append(i)
l
```

Out[64]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [66]:
```
li=[i for i in range(1,11)]
li
```

Out[66]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [72]:
```
# Apply List Comprehension to store the cubes od N natural numbers

li=[i**3for i in range(1,11)]
li
```

Out[72]:
```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [82]:
```
# Function to calculate the factorial

def factorial(n):
    if n==0 or n==1:
        return 1
    return n*factorial(n-1)
#factorial(5)

# Apply list Comprehension to calculate  factorial of a n numbers

n=10
factlist=[factorial(i) for i in range(1,n+1)]
factlist

```

Out[82]:
```
[1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800]
```

In [90]:
```
##### Store cumulative sum of numbers till n in listcomprehension

def cumulsum(n):
    sum=0
    for i in range(1,n+1):
        sum=sum+i
    return sum
n=6
cumulativesum=([cumulsum(i) for i in range(1,n+1)])
cumulativesum

```

Out[90]:
```
[1, 3, 6, 10, 15, 21]
```

In [91]:
```
n=6
cumulativesum=[sum(range(i+1)) for i in range(1,n+1)]
cumulativesum
```

Out[91]:
```
[1, 3, 6, 10, 15, 21]
```

In [96]:
```
# List Compresion to store
# Only leap year is in a given period

st=1970
et=2019
leapyear=[i for i in range(st,et+1) if i%400==0 or (i%100!=0 and i%4==0)]
leapyear
```

Out[96]:
```
[1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016]
```

In [99]:
```
sn=10
en=20
evennumber=[i for i in range(sn,en+1) if i%2==0]
evennumber
```

Out[99]:
```
[10, 12, 14, 16, 18, 20]
```

In [100]:
```
sn=10
en=20
oddnnumber=[i for i in range(sn,en+1) if i%2!=0]
oddnnumber
```

Out[100]:
```
[11, 13, 15, 17, 19]
```

In [ ]:
```

```