In [8]:
```
### Function to print all combinations of pairs of integers in a given list
## [1,2,3] -> (1,2),(1,3),(2,3) -> 3C2 ->3!/((3-2)!)*2!
## [1,2,3,4] -> 1,2  1,3  1,4  2,3  2,4  3,4

## [1,2,3,4] -> 1,2,3  1,2,4  1,3,4  2,3,4

def Combinations(li):
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):
            print(li[i],li[j])
    return
li = [1,2,3]
Combinations(li)
```

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

In [23]:
```

def combinations3(li):
    for i in range(len(li)-1):
        for j in range(i+1,len(li)):
            for k in range(j+1,len(li)):
                print(li[i],li[j],li[k])
    return
li=[1,2,3,4,5]
combinations3(li)
```

```
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
```

In [ ]:

```
## [1,2,3] -> [1,2,3]

def medium(li,k):
    while(True):
        li3 = DifferencePairs(li)
        if li3[0] == li3[1]:
            break
    if len(li3[0])>= k:
        return sorted(li3[0],reverse=True)[k-1]
    else:
        return -1
#     return  li3[0]

### Function to identify all pairs of numbers
### Pairs of numbers and add those differences to the same list
## It returns the updated list and original list

def DifferencePairs(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(str(d))
    li.extend(newelements)
    return [cli,li]
# li=[1,9,8,7,6]
# k=int(input())
# medium(li)
# DifferencePairs(li)
with open('DataFiles/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 [ ]:

```
a = [1,2,3]
b = [1,3,2]
a = b.copy() # Copying the data
a = b[:]
```

In [2]:
```
[4,8]
[20,40,80]
[4,8,12,16]
[3,6,9,12]
# Convert the list into an Arithmetic Progression

[3,8,15]
[3,8,15,5,2,1,4,6,7,9,10,11,12,13,14]

```

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

In [ ]:
```

```

Set - Data Structure in Python

- Represented by '{}'
  - a={1,2,3,4,5,6} --->it contains only unique element and if we kept colon(:) in between the elements means it is dictionary
  - There is no order to the set
  - Dictionary is a 2Dimension type(Keys : Vlaues)
  - Set can contain any kind of data like string, lists etc

In [12]:
```

a={1,1,2,3,4,5}

a.add(7)  #### Adding a single element to the set

# for i in a:
#     print(i,end=' ')  ---> Accessing elements

b = {9,7,8}
li = [11,12,13,1,9]

a.update(b,li)  #### Adding Multiple elements

a

```

Out[12]:
```
{1, 2, 3, 4, 5, 7, 8, 9, 11, 12, 13}
```

In [23]:
```
a = {10,1,2,3,4,5,6}
b = {7,8,9,1,2,3}

a.union(b) ## Union is function which is add the both sets

## A U B = B U A

b.union(a)

```

Out[23]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [27]:
```
a = {10,1,2,3,4,5,6}
b = {7,8,9,1,2,3}
c = {111,123}
a.intersection(b)

```

Out[27]: {1, 2, 3}

In [28]:
```
a.isdisjoint(c)

```

Out[28]: True

In [30]:
```
a - b ## All elements of A which are not in B is equal to A intersection B

b-a

```

Out[30]: {7, 8, 9}

In [33]:
```
g=sorted(a)
g[3]

```

Out[33]: 4

In [34]:
```
a^b   ### Elements either in A or in B

```

Out[34]: {4, 5, 6, 7, 8, 9, 10}

In [35]:
```
### Creating an empty set

d = set()
d

```

Out[35]:
```
set()
```

In [36]:
```

li = [1,2,3,4,2,1,2,3,4]
u = set(li)
u

```

Out[36]:
```
{1, 2, 3, 4}
```

In [ ]:
```

```

Procedural : C

object Oreiented : Java, Python

Scripting : PHP, Python, Javascript, Shell, Perl

Functional : Python, Haskell, Scala

Logic : Prolog, Lisp,

List Comprehensions

In [38]:
```
## List of N natural numbers
n = 10
list = []
for i in range(1,n+1):
    list.append(i)
print(list)

```

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

In [39]:
```
# Another way to add elements to the list
list = [i for i in range(1,11)]
list

```

Out[39]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [41]:
```
## Apply list comprehension to store the cubes on N natural numbers

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

```

Out[41]: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

In [50]:
```
## Function  to calculate the factorial

def Factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * Factorial(n-1)
## Apply list comprehension to calculate factorial of n natural numbers

n = 4
FactorialList = [ Factorial(i) for i in range(1,n+1)]
FactorialList
```

Out[50]: [1, 2, 6, 24]

In [111]:
```
#### Store Cumulative sum of numbers till n
# n = 3, [1, 3, 6]

def csum(n):
    s=0
    for i in range(1,n+1):
        s=s+i

    return  s


```

In [85]:
```
n = 5
sumlist = [sum(range(1,i+1)) for i in range(1,n+1)]
sumlist
```

Out[85]: [1, 3, 6, 10, 15]

In [136]:
```
#### List Comprehension to store  numbers only leap years in a given time
##  start year = 1970  end year = 2019
## leap years = []

a=int(input())
b=int(input())
Leapyears= [i for i in range(a,b+1) if (i%400==0 or (i%100!=0 and i%4==0))]
Leapyears

```

```
1970
2019
```

Out[136]: [1972, 1976, 1980, 1984, 1988, 1992, 1996, 2000, 2004, 2008, 2012, 2016]

In [138]:
```
## EVEN NUMBERS
n = [i for i in range(1,10) if(i%2==0)]
n
```

Out[138]: [2, 4, 6, 8]

In [139]:
```
## ODD NUMBERS
n = [i for i in range(1,10) if(i%2!=0)]
n
```

Out[139]: [1, 3, 5, 7, 9]

In [147]:
```

li = [1,2,3,2,1]
li.sort()
u2 =[]
u1 = []
u3 = [li[i] for i in range(0,len(li)-1) if sorted(li)[i]!=sorted(li[i+1]) an
# u1= [u2.append(i) for i in li if i not in u2]
u2

```

Out[147]: [1, 2, 3]

In [ ]:
```


```