In [1]:

```python
# Print First N natural numbers.
def print_1_to_n(n):
    if n == 0:
        return
    print_1_to_n(n-1)
    print(n)

print_1_to_n(5)
```

```
1
2
3
4
5
```

In [2]:

```python
# Print First N natural numbers in reverse order.
def print_n_to_1(n):
    if n == 0:
        return
    print(n)
    print_n_to_1(n-1)

print_n_to_1(5)
```

```
5
4
3
2
1
```

In [3]:

```python
# Check list is sorted or not

def isSorted(a):
    l = len(a)
    if l==0 or l==1:
        return True

    if a[0]>a[1]:
        return False

    smallerList = a[1:]
    isSmallerListSorted = isSorted(smallerList)

    if isSmallerListSorted:
        return True
    else:
        return False
```

In [4]:

```
a = [1,2,3,4,5,6]
isSorted(a)
```

Out[4]:

```
True
```

In [5]:

```
# Check list is sorted or not in better way

def isSortedBetter(a,si):
    l = len(a)
    if si == l-1 or si == l:
        return True
    if a[si]>a[si+1]:
        return False
    isSmallerPartSorted = isSortedBetter(a,si+1)
    return isSmallerPartSorted

isSortedBetter([1,2,3,44,5],0)
```

Out[5]:

```
False
```

In [6]:

```
# Sum of Array recursively

def sum_of_Array(a,N):
    if N<=0:
        return 0

    return sum_of_Array(a,N-1) + a[N-1]
```

In [7]:

```
sum_of_Array([3,1,5],3)
```

Out[7]:

```
9
```

In [12]:

```
# Getting Index of the first occurance of an element

def get_index(a,x):
    l = len(a)
    if l == 0:
        return -1

    if a[0]==x:
        return 0

    smallerList = a[1:]
    smallerListOutput = get_index(smallerList, x)

    if smallerListOutput == -1:
        return -1
    else:
        return smallerListOutput + 1
```

In [ ]:

```
def firstIndex(a,x):
    l = len(a)
    if l == 0:
        return -1

    if a[0] = x:
        return 0

    smallerList = a[1:]
    smallerListOutput = firstIndex(smallerList, x)

    if smallerListOutput == -1:
        return -1
    else:
        return smallerListOutput + 1
```

In [13]:

```
get_index([2,3,3,4,4],4)
```

Out[13]:

```
3
```

In [27]:

```
def get_index_better(a,x,si):
    l = len(a)
    if si == l:
        return -1
    if a[si] == x:
        return si
    smallerList = get_index_better(a,x,si+1)
    if smallerList==-1:
        return -1
    else:
        return smallerList
```

In [30]:

```
def firstIndexBetter(a,x,si):
    l = len(a)

    if si == l:
        return -1

    if a[si] == x:
        return si
    smallerList = firstIndexBetter(a,x,si+1)
    if smallerList == -1:
        return -1
    else:
        return smallerList
```

In [31]:

```
firstIndexBetter([2,3,22,4,22,4],222,0)
```

Out[31]:

```
-1
```

In [39]:

```python
# Getting last Index of an element

def lastIndex(a,x):
    l = len(a)
    if l == 0:
        return -1

    smallerList = a[1:]
    smallerListOutput = lastIndex(smallerList, x)



    if smallerListOutput != -1:
         return smallerListOutput + 1
    else:
        if a[0] == x:
            return 0
        else:
            return -1

```

In [40]:

```python
lastIndex([1,2,1],1)
```

Out[40]:

```
2
```

In [3]:

```python
# Replace Char in a string Recursively

def replaceChar(s,a,b):
    if len(s) == 0:
        return s

    smallOutput = replaceChar(s[1:],a,b)
    if s[0] == a:
        return b + smallOutput
    else:
        return s[0] + smallOutput
```

In [4]:

```python
replaceChar("cbc","c","A")
```

Out[4]:

```
'AbA'
```

In [5]:

```python
def replaceChar(s,a,b):
    if len(s) == 0:
        return s

    small = replaceChar(s[1:],a,b)
    if s[0] == a:
        return b + small
    else:
        return s[0] + small
```

In [6]:

```python
replaceChar("aba","a","0")
```

Out[6]:

```
'0b0'
```

In [12]:

```python
string="geeksforgeeks"
p=""
for char in string:
    if char not in p:
        p = p + char
print(p)
k=list("geeksforgeeks")

```

```
['s', 's', 's', 's', 's', 's', 's', 's', 's', 's', 's']
```

In [13]:

```python
# Binary Search Recursively

def binarySearch(a,x,si,ei):

    if si>ei:
        return -1

    mid = (si+ei)//2
    if a[mid] == x:
        return mid
    elif a[mid]>x:
        return binarySearch(a,x,si,mid-1)
    else:
        return binarySearch(a,x,mid+1,ei)
```

In [14]:

```python
binarySearch([2,3,4,5,6],3,0,4)
```

Out[14]:

```
1
```

Merge Sort Recursively

In [12]:

```
def merge(a1,a2,a):
    i = 0
    j = 0
    k = 0
    while i<len(a1) and j<len(a2):
        if (a1[i]<a2[j]):
            a[k] = a1[i]
            k = k+1
            i = i+1
        else:
            a[k] = a2[j]
            k = k+1
            j = j+1
    while i<len(a1):
            a[k] = a1[i]
            i = i+1
            k = k+1
    while j<len(a2):
            a[k] = a2[j]
            j = j+1
            k = k+1
```

In [13]:

```
def merge_sort(a):
    if len(a) == 0 or len(a) == 1:
        return
    mid = len(a)//2
    a1 = a[:mid]
    a2 = a[mid:]

    merge_sort(a1)
    merge_sort(a2)

    merge(a1,a2,a)
```

In [14]:

```
a = [10,5,3,1,7,9,4]
merge_sort(a)
a
```

Out[14]:

```
[1, 3, 4, 5, 7, 9, 10]
```

In [ ]:

```

```

In [ ]:

```

```