Circular Counter

```
a = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
def circ_c oun ter (in t_list, skip):
       skip = skip - 1 # list starts with 0 index
       idx = 0
        while len(in t_list) > 0:
               # hashing to keep changing the
index to every 3rd
               idx = (skip + idx) % len(in t_list)
                pri nt( int _li st.p op (idx))
circ_c oun ter(a, 3)
```

Flatten Array

```
from functools import reduce
def list_f lat ten(l):
       def flatte n(pv, cv):
               if not isinst anc e(cv, list):
                       return pv + [cv]
                else:
                       return pv + list_f lat -
ten(cv)
        return reduce (fl atten, l, [])
print( lis t_f lat ten([2, 1, [3, [4, 5], 6], 7,
[8]]))
print( 'output should be [2, 1, 3, 4, 5, 6, 7,
8]')
```

Three Sum Array

```
# TODO: FFT implementation for MlogM algo
def three_ sum (nums):
       res = []
        num s.s ort()
       for i in range( len (nums) - 2):
               if i > 0 and nums[i] == nums[i - 1]:
                       con tinue
               l, r = i + 1, len(nums) - 1
                while l < r:
                      s = nums[i] + nums[l] +
nums[r]
                      if s > 0:
                              r -= 1
                      elif s < 0:
```

Three Sum Array (cont)

```
>              l += 1
       else:
          # found three sum
          res.append([nums[i], nums[l], nums[r]])
          # remove duplicates
          while l < r and nums[l] == nums[l + 1]:
            l += 1
          while l < r and nums[r] == nums[r - 1]:
            r -= 1
          l += 1
          r -= 1
   return res
if __name__ == "__main__":
   x = [-1, 0, 1, 2, -1, -4]
   print("input: ", x)
   print("output should be: ", [[-1, -1, 2], [-1, 0, 1]])
   print("output: ", three_sum(x))
```

Summary Ranges

```
def summary_ranges(nums):
       res = []
       l = len(nums)
       if l == 1:
                return [str(n ums [0])]
       i = 0
        while i < l:
                start = nums[i]
                while i + 1 < l and nums[i + 1] -
nums[i] == 1:
                       i += 1
               if nums[i] != start:
                        res.ap pen d(s tr( start) +
" ->" + str(nu ms[i]))
                else:
                        res.ap pen d(s tr( start))
               i += 1
        return res
a = [0, 1, 2, 4, 5, 7]
```

Summary Ranges (cont)

```
> print("input: ")
print(a)
print("output should be")
print(["0->2", "4->5", "7"])
print("output: ")
print(summary_ranges(a))
```

Rotate Arrays

```
def rotate(arr, k):
       n = len(arr)
       k = k % n
        return arr[n - k:] + arr[:n - k]
a = [1, 2, 3, 4, 5, 6, 7]
print( "in: ", a)
print( " exp ected: ", [5, 6, 7, 1, 2, 3, 4])
print( "out: ", rotate(a, 3))
```

Longest Non Repeat

```
from typing import Dict
def longes t_n on_ rep eat(s: str) -> int:
        start, maxlen = 0, 0
        use d_char = {} # type: Dict[str, int]
       for i, char in enumer ate(s):
              if char in used_char and start <=
used_c har [char]:
                      start = used_c har [char] +
1
               else:
                      maxlen = max(ma xlen, i -
start + 1)
               use d_c har [char] = i
        return maxlen
a = " abc abc def bb"
print( " input: ", a)
print( " result: ", longes t_n on_ rep eat(a))
print( " output should be 6")
```

Missing Ranges

```
def missing_ranges(arr, low, high):
        hashed = set(arr)
       for n in range(low, high):
              if n not in hashed:
                     pri nt(n)
inpt = [10, 12, 11, 15]
low, hi = 10, 15
print( " input: ", inpt)
print( " result: ")
missin g_r ang es( inpt, low, hi)
```