Linear Array

```
def iteration(source):
  for i in range(len(source)):
    print(source[i])

def reverseIteration(source):
  for i in range(len(source) - 1, -1, -1):
    print(source[i])
```

```
def copyArray(source):
  newArray = [None] * len(source)
  for i in range(len(source)):
    newArray[i] = source[i]
  return newArray
```

```
def resizeArray(oldArray, newCapacity):
  newArray = [None] * newCapacity
  for i in range(len(oldArray)):
    newArray[i] = oldArray[i]
  return newArray
```

```
def shiftLeft(arr):
  for i in range(1, len(arr)):
    arr[i-1] = arr[i]
  arr[len(arr) - 1] = None
  return arr
```

```
def shiftRight(arr):
  for i in range(len(arr) - 1, 0, -1):
    arr[i] = arr[i - 1]
  arr[0] = None
  return arr
```

```
def insertElement(arr, size, elem, index):
  # Practice how to throw exception if there
  if size == len(arr):
    print("No space left. Insertion failed")
  else:
    for i in range(size, index, -1):
      arr[i] = arr[i - 1] #Shifting right ti
    arr[index] = elem #Inserting element
    return arr
```

```
def removeElement(arr, index, size):
  for i in range(index + 1, size):
    arr[i - 1] = arr[i] #Shifting left from
  arr[size - 1] = None #Making last space em
```

```
def rotateLeft(arr):
  temp = arr[0]
  for i in range(1, len(arr)):
    arr[i-1] = arr[i]
  arr[len(arr) - 1] = temp
  return arr
```

```
def rotateRight(arr):
  temp = arr[len(arr) - 1]
  for i in range(len(arr) - 1, 0, -1):
    arr[i] = arr[i - 1]
  arr[0] = temp
  return arr
```

Circular Array

```
# Forward Iteration
def forwardIteration(cir, start, size):
  k = start
  for i in range(size):
    print(cir[k])
    k = (k + 1) % len(cir)

# Backward Iteration
def backwardIteration(cir, start, size):
  k = (start + size - 1) % len(cir)
  for i in range(size):
    print(cir[k])
    k = k - 1
    if k == -1:
      k = len(cir) - 1
```

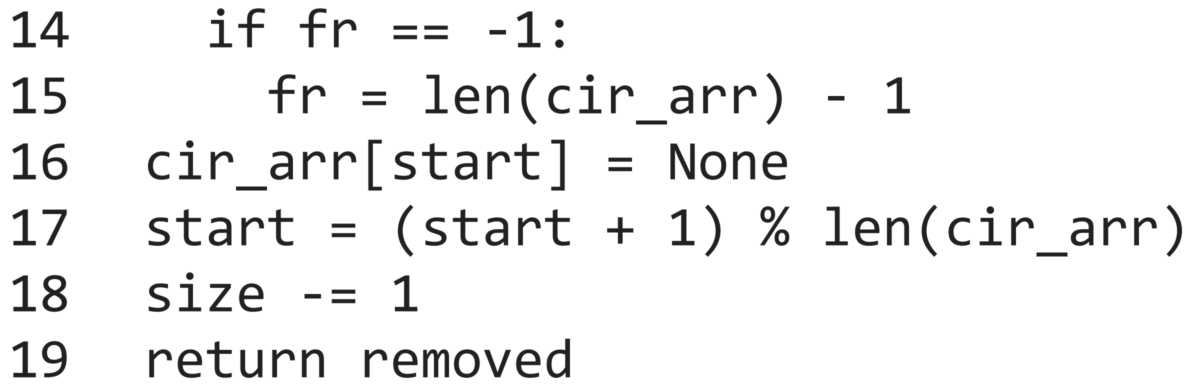
```
# Linearizing Circular Array
def linearizingCircularArray(cir_arr, size,
  lin_arr = [None] * size # Initializing wit
  k = start
  for i in range(size):
    lin_arr[i] = cir_arr[k]
    k = (k + 1) % len(cir_arr)
  return lin_arr
```

```
# Resizing Circular Array
def resizingCircularArray(cir_arr, start, si
  new_arr = [None] * new_capacity
  k = start
  for i in range(size):
    new_arr[i] = cir_arr[k]
    k = (k + 1) % len(cir_arr)
  return new_arr
```

```
# Insert in Circular Array
def insert(cir_arr, start, size, elem, pos):
  if size == len(cir_arr):
    cir_arr = resizingCircularArray(cir_arr,
  nShifts = size - pos
  fr = (start + size - 1) % len(cir_arr)
  to = (fr + 1) % len(cir_arr)
  for i in range(nShifts):
    cir_arr[to] = cir_arr[fr]
    to = fr
    fr = fr - 1
    if fr == -1:
      fr = len(cir_arr) - 1
  idx = (start + pos) % len(cir_arr)
  cir_arr[idx] = elem
  size += 1
```

```
# Remove value from circular array by left s
def removeByLeftShift(cir_arr, start, size,
  nShift = size - pos - 1
  idx = (start + pos) % len(cir_arr)
  removed = cir_arr[idx]
  to = idx
  fr = (to + 1) % len(cir_arr)
  for i in range(nShifts):
    cir_arr[to] = cir_arr[fr]
    to = fr
    fr = (fr + 1) % len(cir_arr)
  cir_arr[fr] = None
  size -= 1
  return removed
```

```
# Remove value from circular array by right
def removeByRightShift(cir_arr, start, size,
  nShift = pos
  idx = (start + pos) % len(cir_arr)
  removed = cir_arr[idx]
  to = idx
  fr = (to - 1)
  if fr == -1:
    fr = len(cir_arr) - 1
  for i in range(nShifts):
    cir_arr[to] = cir_arr[fr]
    to = fr
    fr -= 1
```



```
  if fr == -1:
    fr = len(cir_arr) - 1
cir_arr[start] = None
start = (start + 1) % len(cir_arr)
size -= 1
return removed
```