تمرین چهارم

```
class Node:
  def init(self, coef, power):
    self.coef = coef
    self.power = power
    self.prev = None
    self.next = None

class List:
  def init(self):
    self.head = Node(None, None)
    self.head.next = self.head
    self.head.prev = self.head
    self.n = 0

  def insert_after(self, x, coef, power):
    # We assume that the caller of this method ensures 'x' is a valid node within the list.
    y = Node(coef, power)
    y.prev = x
    y.next = x.next
    x.next.prev = y
    x.next = y
    self.n += 1
    return y

  def insert(self, coef, power):
    # Automatically places the new node in descending power order
    x = self.head.next
    while x != self.head and x.power > power:
      x = x.next
    self.insert_after(x.prev, coef, power)

  def node_at(self, ind):
    if ind < 0 or ind >= self.n:
      raise Exception("Index out of bounds")
    x = self.head.next
    for i in range(ind):
      x = x.next
    return x

  def get(self, ind):
    # Uses node_at to simplify operation
    x = self.node_at(ind)
    return f"coef : {x.coef}, power : {x.power}"

  def delete(self, ind):
    # Simplified delete method using indices
    x = self.node_at(ind)
    x.prev.next = x.next
    x.next.prev = x.prev
    self.n -= 1
    return x

  def size(self):
    return self.n

  def add(self, ind1, ind2):
    node1 = self.node_at(ind1)
    node2 = self.node_at(ind2)
    node1.coef += node2.coef
    self.delete(ind2)

  def mul(self, coef1, power1, coef2, power2):
    node1 = self.find(coef1, power1)
    node2 = self.find(coef2, power2)
    if node1.power == node2.power:
      result_coef = node1.coef * node2.coef
      result_power = node1.power
      self.delete(node1)
      self.delete(node2)
      self.insert(result_coef, result_power)
    elif node1.coef == node2.coef:
      result_coef = node1.coef
      result_power = node1.power + node2.power
      self.delete(node1)
      self.delete(node2)
      self.insert(result_coef, result_power)
    else:
      raise Exception("multiply operation can't be done!")
```