```
print("WELCOME! TO DATA SCIENCE") # Kurt Russel Villamor CPE22S3
```

```
WELCOME! TO DATA SCIENCE
```

```
"""SINGLY""" # Traversing only forward
class Node:
  def __init__(self, data=None, next = None):
    self.data = data
    self.next = next

  def add_node(self, node):
    if self.next is None:
      self.next = node
    else:
      while self is not None:
        if self.next is None:
          self.next = node
          break
        self = self.next

  def delete_node(self, data):
    if  self.data == data:
      if self.next is None:
        self.data = None
      else:
          self.data = self.next.data
          self.next = self.next.next
    else:
      while self is not None:
        if self.next is None:
          pass
        else:
          if self.next.data == data:
            self.next = self.next.next
        self = self.next

  def print_nodes(self):
    while self is not None:
      print(self.data)
      self = self.next


```

```
# Nodes can be added Different Approaches
root = Node(data = 'C')
root.next=Node(data ='P')
root.add_node(Node(data ='E'))
```

```
root.print_nodes() # Print Nodes
```

```
C
P
E
```

```
root.delete_node('P') # Deletes The Middle Node in C-P-E
```

```
root.print_nodes() # Successfully Deletes P
```

```
C
E
```

```
root.delete_node('C' ) # Deletes The Root Node in C-P-E
```

```
root.print_nodes() # Successfully Deletes C
```

```
E
```

```
root.delete_node('E') # Deletes The Root Node/Last Node in C-P-E
```

```
root.print_nodes() # Successfully Deletes E and Returns None
```

```
    None
```

```
"""DOUBLY"""  # Traversing forward and Backward
class DNode:
  def __init__(self, data, next = None, prev = None):
    self.data = data
    self.next = next
    self.prev = prev

  def add_node(self, node):
    if self.next is None:
      self.next = node
      node.prev = self
    else:
      while self is not None:
        if self.next is None:
          self.next = node
          node.prev = self
          break
        self = self.next

  def delete_node(self, data):
      while self is not None:
        if self.next is not None:
          if self.next.data == data:
            self.next = self.next.next
            self.next.prev = self
        self = self.next


  def print_nodes(self):
    while self is not None:
      print(self.data)
      self = self.next
```

```
# Nodes can be added Different Approaches
Droot = DNode(data = 'C')
Droot2 = DNode(data ='P')
Droot.add_node(Droot2)
Droot.add_node(DNode(data ='E'))
```

```
Droot.print_nodes()   # Print Nodes
```

```
    C
    P
    E
```

```
print(Droot2.prev.data)  # Print the previous data of P
```

```
    C
```

```
print(Droot.prev)  # Print the previous data of C
```

```
    None
```

```
Droot.delete_node('P')  # Deletes the Middle Node in CPE
```

```
Droot.print_nodes() # Succesfully Deletes P
```

```
    C
    E
```

```
print(Droot.next.data) # Printing E
```

```
    E
```

```
trial = Droot.next #  Setting trial for checking E
```

```
print(trial.prev.data) # prints the Previous of E
```

```
C
```

```
"""CIRCULAR"""  # Traversing forward and Backward and head is also Tail
class DNode:
  def __init__(self, data, next = None, prev = None):
    self.data = data
    self.next = next
    self.prev = prev

  def add_node(self, node):
    if self.next is None:
      self.next = node
      node.prev = self
    else:
      while self is not None:
        if self.next is None:
          self.next = node
          node.prev = self
          break
        self = self.next

  def delete_node(self, data):
      while self is not None:
        if self.next is not None:
          if self.next.data == data:
            self.next = self.next.next
            self.next.prev = self
        self = self.next


  def print_nodes(self):
    while self is not None:
      print(self.data)
      self = self.next
```

```
# Nodes can be added Different Approaches
Droot = DNode(data = 'C')
Droot2 = DNode(data ='P')
Droot.add_node(Droot2)
Droot.add_node(DNode(data ='E'))
```

```
print(Droot2.next.next)
```

```
None
```