1. Circular Single Linked List

```
class Node:
  key
  data
  next

  constructor Node(k, d):
    key = k
    data = d
    next = null

class CircularLinkedList:
  Head
  Tail

  constructor CircularLinkedList(k, d):
    Head = new Node(k, d)
    Tail = Head

  procedure insertAtEnd(k, d):
    newEnd = new Node(k, d)
    newEnd.next = Head
    Tail.next = newEnd
    Tail = newEnd

  procedure printList():
    current = Head
    while current != null:
        print(current.data, "->", end="")
        if current.next == Head:
            print(current.next.data)
            break
        current = current.next

procedure insertAtFirst(k, d):
    newHead = new Node(k, d)
    Tail.next = newHead
    newHead.next = Head
    Head = newHead

procedure insertAtMid(key, d):
    current = Head
    while current.next != Head:
        if current.key == key:
            print("Node Already Exists. Select a different key.")
            break
        else:
            if current.key < key and current.next.key > key:
                newMid = new Node(key, d)
                newMid.next = current.next
                current.next = newMid
                break
            else:
                current = current.next

procedure deleteNode(key):
    delNode = Head
```

```
temp = null
while delNode != null:
  if delNode.key == key and delNode == Head:
    Head = delNode.next
    Tail.next = Head
    delete delNode
    break
  else:
    if delNode.key == key:
      temp.next = delNode.next
      delete delNode
      break
    temp = delNode
    delNode = delNode.next
```

2. Doubly Linked List

```
class Node:
  key
  data
  next
  prev

  constructor Node(k, d):
    key = k
    data = d
    next = null
    prev = null

class DoublyLinkedList:
  Head
  Tail

  constructor DoublyLinkedList(k, d):
    Head = new Node(k, d)
    Tail = Head

  procedure insertAtEnd(k, d):
    newNode = new Node(k, d)
    newNode.next = null
    Tail.next = newNode
    newNode.prev = Tail
    Tail = newNode
```

```
procedure insertAtHead(k, d):
   newNode = new Node(k, d)
   newNode.prev = null
   newNode.next = Head
   Head = newNode

procedure insertInBetween(key, d):
   newNode = new Node(key, d)
   current = Head
   while current != null:
      if current.key == key:
         print("Node Already Exists. Select a different key.")
         break
      else:
         if current.key < key and current.next.key > key:
            newNode.prev = current
            newNode.next = current.next
            current.next = newNode
            break
         else:
            current = current.next

procedure deleteNode(key):
   delNode = Head
   while delNode != null:
      if delNode.key == key and delNode == Head:
         Head = delNode.next
         delete delNode
         break
```

```
        else:
            if delNode.key == key:
                delNode.prev.next = delNode.next
                delNode delete delNode
            else:
                delNode = delNode.next

procedure printListForward():
    printNode = Head
    print("NULL -> ", end="")
    while printNode != null:
        print(printNode.data, " -> ", end="")
        printNode = printNode.next
    print("NULL")

procedure printListBackwards():
    printNode = Tail
    print("NULL -> ", end="")
    while printNode != null:
        print(printNode.data, " -> ", end="")
        printNode = printNode.prev
    print("NULL")
```

3. Circular Doubly Linked List

```
class Node:
  key
  data
  next
  prev

  constructor Node(k, d):
    key = k
    data = d
    next = null
    prev = null

class DoublyCircularLinkedList:
  Head
  Tail

  constructor DoublyCircularLinkedList(k, d):
    Head = new Node(k, d)
    Tail = Head

  procedure insertAtEnd(k, d):
    newNode = new Node(k, d)
    Tail.next = newNode
    newNode.prev = Tail
    Tail = newNode
    Tail.next = Head
    Head.prev = Tail
```

```
procedure insertAtFront(k, d):
    newNode = new Node(k, d)
    Tail.next = newNode
    newNode.prev = Tail
    newNode.next = Head
    Head.prev = newNode
    Head = newNode

procedure insertBetween(key, d):
    newNode = new Node(key, d)
    current = Head
    while current != null:
        if current.key == key:
            print("Key Exists. Insert a node with a different key.")
            break
        else:
            if current.key < key and current.next.key > key:
                newNode.prev = current
                newNode.next = current.next
                current.next.prev = newNode
                current.next = newNode
                break
            current = current.next

procedure deleteNode(key):
    delNode = Head
    while delNode != null:
        if delNode.key == key and delNode == Head:
            Tail.next = delNode.next
            delNode.next.prev = Tail
            delete delNode
            break
        else:
            if delNode.key == key:
                delNode.prev.next = delNode.next
                delNode.next.prev = delNode.prev
                delete delNode
                break
            delNode = delNode.next

procedure printForward():
    printNode = Head
    while printNode != null:
        print(printNode.data, " -> ", end="")
        if printNode.next == Head:
            print(printNode.next.data)
            break
        printNode = printNode.next

procedure printBackwards():
    printNode = Tail
    while printNode != null:
        print(printNode.data, " -> ", end="")
        if printNode.prev == Tail:
            print(printNode.prev.data)
            break
        printNode = printNode.prev
```