שעור 12 רשימות מקושרות – Linked Lists המשך

רשימה מקושרת ממוינתSorted Linked List pseudo-code

```
Class Node
   Object data,
   Node next
// constructor
Node(newData, nextNode)
       data = newData
       next = nextNode
end-constructor
End-Node
End-Class-Node

Class SortedList
   Node head, tail
int size
// constructor
SortedList()
      head = tail = null
      size = 0
End-SortedList()

Add(newData)
      if (head == null) then
node = new Node(newData, null)
      head = tail = node
       else if (newData<head.data) then
         node = new Node(newData, head)
         head = node
else
          Node temp = head, prev = head
// find place
          while(temp ≠ null and temp.data<newData)
prev = temp
temp = temp.next
          end-while
// add after tail
          if (temp == null) then
node = new Node(newData, null)
tail.next = node
             tail = node
```

```
// add in the middle
         else
node = new Node(newData, temp)
prev.next = node
end-if
end-if
size = size + 1;
End-Add

Remove(data)
if (head == null) return null
   if (head.data == data) then
      head = head.next
      size = size - 1
      return data
   end-if
Node temp = head, prev = head
// find place
while(temp.next ≠ null and temp.data<data)
prev = temp
temp = temp.next
   end-while
// remove tail
if (temp.next == null and temp.data == data) then
prev.next = null
      size = size - 1
      return data
// remove middle element
if (temp.next!= null and temp.data == data) then
prev.next = temp.next
      size = size - 1
      return data
 end-if
// element not found
 return null
End-Remove
```

רשימה מקושרת מעגלית pseudo-codeCycle Linked List

```
Class CycledList
   Node head, tail
int size
// constructor
CycledList()
      head = tail = null
      size = 0
End-CycledList()

Add(newData)
if (head == null) then
      head = new Node(newData, null)
head.next = head
       else
node = new Node(newData, head)
tail.next = node
           tail = node
end-if
       size = size + 1
 end-Add
Remove(data)
if (head == null) return null
   if (head.data == data) then
      head = head.next
tail.next = head
      size = size - 1
      return data
   end-if
Node temp = head, prev = head
// find place
   while (temp.next ≠ head and temp.data ≠ data)
prev = temp
temp = nemp.next
   end-while
// remove tail
   if (temp.next == head and temp.data== data)then
prev.next = head
size = size - 1
return data
   end-if
```

```
// remove middle element
if (temp.data == data) then
prev.next = temp.next
size = size - 1
return data
   end-if
   return null
end-Remove
```