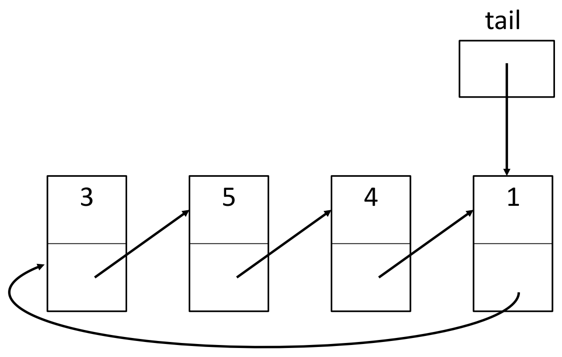
Module IN2002—Data Structures and Algorithms
Answers to Exercise Sheet 5

1. Work out (using pictures) what the following procedure does on circular lists:

```
public void modify(CList list) {
     if (! isEmpty()) {
          Node tmp = tail.next;
          tail.next = tail.next.next;
          tmp.next = tmp;
     }
}
```

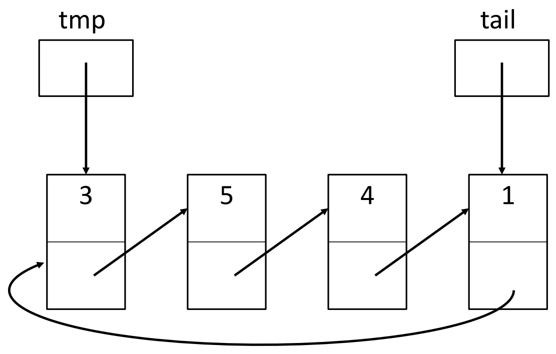
Consider the cases where the list has more than one node, only one node, or no nodes.

- First we create a circular list with random information to test the code on:

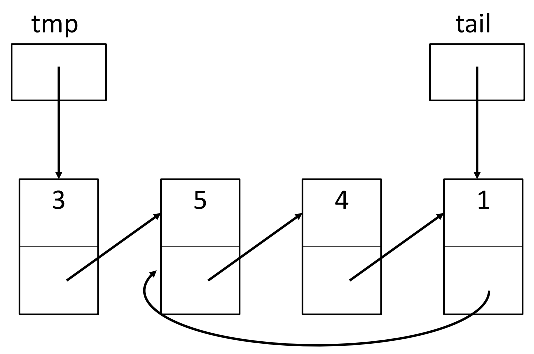


`if (! isEmpty())` checks that the list has at least one node

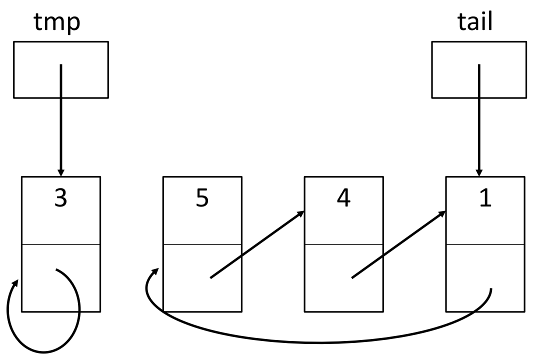
`Node tmp = tail.next;`



`tail.next = tail.next.next;`

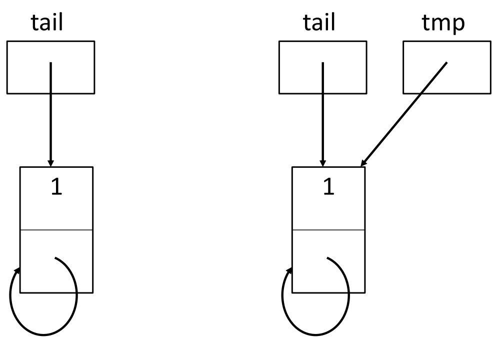


```
tmp.next = tmp;
```



- So the function removes the head from the list, and creates a new circular list containing only the former head. However, this new circular list is not returned by the function.

- With only one node, nothing happens:



- With no nodes, nothing happens.

2. Write functions that return the maximum of the numbers of (where the info in each node is a number):

a) A circular list

```
int findmax() {
   if (! notEmpty()) {
      node p = tail.next;
      int max = p.info;
      while (p != tail) {
         p = p.next;
         if (p.info > max)
              max = p.info;
      }
      return max;
   } else
         return Integer.MIN_VALUE;
         // This value indicates the empty list.
}
```

b) A doubly linked list

```
int findmax() {
     if (! notEmpty()) {
          node p = head;
          int max = p.info;
          while (p != tail) {
               p = p.next;
               if (p.info > max)
                    max = p.info;
          }
          return max;
     } else
          return Integer.MIN_VALUE;
          // This value indicates the empty list.
}
```

3. Write a procedure to swap the first two nodes of a doubly linked list.
Have you covered all the cases?

- We first want to check that the list has at least two nodes. A convenient test is to compare head and tail If the list has only one node, they will be equal. If the list is empty, they will both be null, and therefore also equal. If the list has at least two nodes, the two pointers will refer to different nodes.

- The simplest thing is just to swap the info components of the first two nodes (provided there are at least two):

```
void swapFirstTwo() {
     if (head != tail) {
          int tmp = head.infor;
          head.info = head.next.info;
          head.next.info = tmp;
     }
}
```

- But the intention of the question was an exercise in pointer manipulation. An easy method is to introduce a pointer to the node after the head, and update head at the very end:

```
void swapFirstTwo() {
     if (head != tail) {
          DLLNode neck = head.next;
          head.next = neck.next;
          neck.next = head;
          head.prev = neck;
          neck.prev = null;
          if (tail == neck) // two element list
               tail = head;
          head = neck;
     }
}
```

- Note that if the second node is also the last one, tail must be set to the old first node.