Disc 1

Fundamental Data Structure

Singly linked list:
- A collection of nodes in order linear, you can move in 1 direction from node to node.

```
head
| a | b | c | d | fnull |
  |   |   |   |   |
  |   |   |   |   |
  |   |   |   |   |
next
```

R 3.6 algorithm findSecondToLast ()
```
if (size<2) then
    return throw new Exception

Node n = head
while (n != tail)
    n = n.next()
return n
```

R 3.9 int size ()
```
int c = 0; Node n = head;
while (n != null)
    c++;
    n = n.next();
return c;
```

C 3.35 algorithm concatenate (L, M)
```
L = L.next (M.head())
```

algorithm concatenate (L, M)
```
Node n = M.head()
while (n.next() != null)
    n = n.next()
    n.setNext (H.getHead())
L = L
```
Doubly Linked List:

- Each node has 2 references (next, previous).
- It has 2 sentinel nodes (header, trailer).

R3.8) Node middle ( )

Node n = header.getHead ( )
Node m = trailer.getPrev ( )
if (n == trailer) return null
while (n != m) {
    n = n.getNext ( )
    m = m.getNext ( )
}
return m, n

S.26) Algorithm concatenate ( L, M )

Node n = ( L.getTrailer ( ) ).getPrev ( )
Node m = ( M.getHead ( ) ).getNext ( )
(M.getHead ( )).setNext (null);
(L.getTrailer ( )).setPrev (null);
n.setNext (m);
m.setPrev (n);
L = L;
return L ;
in single linked list

algorithm swap_S(x, y)
  m ← y.getNext()
  node n ← head
  if n = x then
    x.setNext(y.prev)
    y.setNext(x)
  else
    while (n getNext () = x)
      n ← n.getNext()
      n.setNext(y)
      y.setNext(x)
      x.setNext(m)

in doubly linked list

algorithm swap_D(x, y)
  node n ← x.getPrev()
  node m ← y.getPrev()
  n.setNext(y)
  y.setPrev(n)
  y.setNext(x)
  x.setPrev(y)
  m.setPrev(x)
Circularly LinkedList

- There is no head reference.
- There is only tail reference & the rest of the tail back to tail.

R3.10

```c
int size() {
    int c = 0;
    Node* n = tail.getNext();
    while (n != tail) {
        c++;
        n = n.getNext();
    }
    return c;
}
```