Cairo University
Faculty of Computers and Artificial Intelligence
Computer Science Department

Data structures
CS214

Linked lists

Dr. Amin Allam

1 Linked lists

A linked list is a container which is intended to store any number of elements of the same type and maintains them in a specific linear ordering. Contrary to arrays, lists generally allow fast insertion and deletion of elements with low memory overhead. However, list elements are fragmented in memory which increases cache misses.

A linked list object consists of:
- A pointer (usually called head) that holds the address of the first node in the list.
- A sequence of nodes. Each node consists of:
  - The data associated of an element contained in the list.
  - A pointer holding the address of the next node in the list.

A linked list object may also contain:
- A pointer (usually called tail) that holds the address of the last node in the list.
- An integer that maintains the number of elements in the list.

The head is mandatory and allows access to all elements in the list. The tail is optional and allows fast insertion at the end of the linked list.

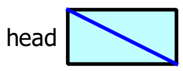
The examples in this document are restricted to linked lists of integer elements. However, linked lists may contain elements of any type, such that all elements in the same linked list object must be of the same type.

Most examples are concerned with linked lists which contain head only (does not contain tail). However, they can be easily modified to support tail by just updating tail if the last node changes.

2 Insertion at the beginning of a linked list

1

```
head = 0;
```

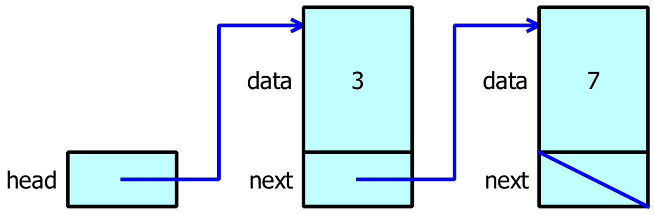


|< < > >|

3 Insertion at the end of a linked list which does not include tail

1

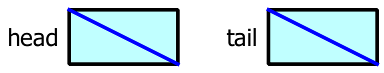
```
// If head is not 0:
```



|< < > >|

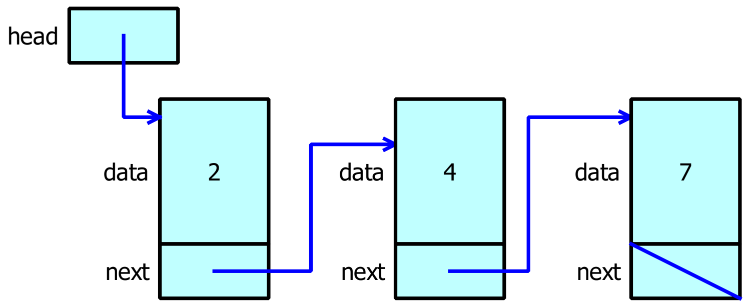
4 Insertion at the end of a linked list which includes tail

1
```
head = tail = 0;
```



5 Insertion in an ordered linked list

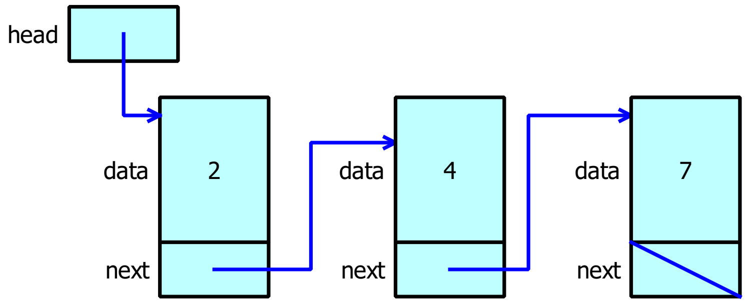
1
```
// If head is not 0
// and head->data<=5:
```



6 Insertion in an ordered linked list (another version)

$\boxed{1}$

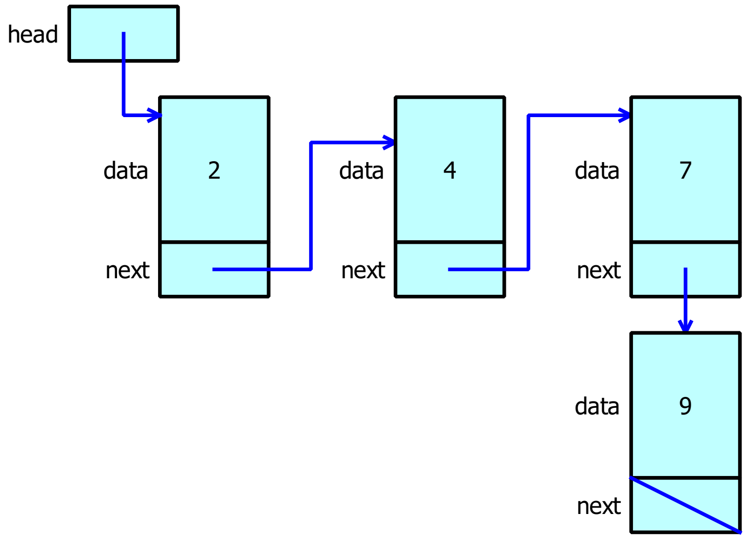
```
// If head is not 0
// and head->data<=5:
```



7 Deletion from a linked list

$\boxed{1}$

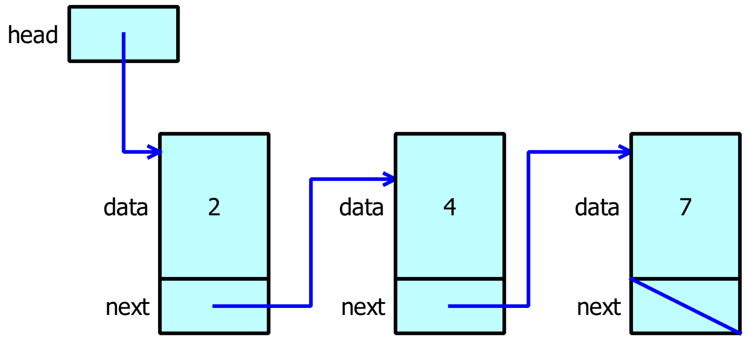
```
// If head is not 0
// and head->data!=7:
```



8 Reversing a linked list

1

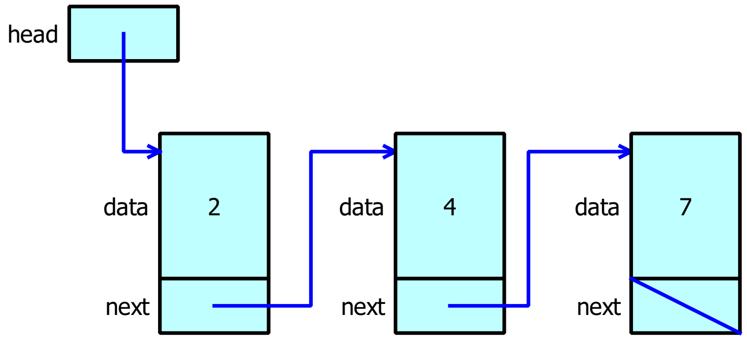
```
// If head is not 0:
```



9 Reversing a linked list recursively

1

```
if(head != 0)
```


head
data
2
next
data
4
next
data
7
next