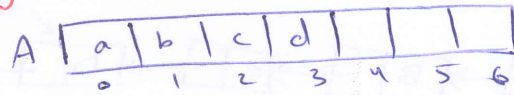


Chapter 3 Arrays and linked lists.

3.1 Using Arrays



⊗ مجموعة من elements لا نفس ال type

⊗ كل cell ال index منه بقدر الوصول اليه من خلال وضع ال cells هياكل بقوى ال عناصر
⊗ كل array لا يملك ال length عدد مسبقاً جدد قد يثبت بقدر اخذ عناصر فيه (capacity)

⊗ $A[k]$: يبين العنصر (element) ال بي ال index
 $A(2) = c$ مثال

⊗ بقدر اخذ عناصر او references و objects منه.

⊗ Declaring Arrays ⊗

Type [] name = new Type [length]
↓ ex

GameEntry [] entries = new GameEntry [maxEntries];

Type [] name = { elements }
↓ java type or class name

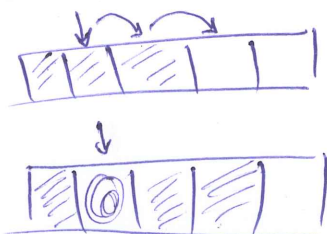
int [] sample = { 1, 2, 3 }
↓ ex

⇒ Insertion

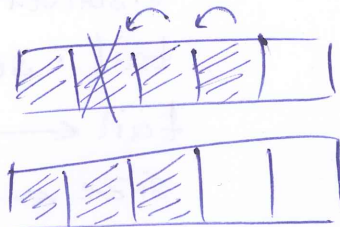
الفكرة بالاضافة الي افض مكان للعنصر ال بي اضيفه عن طريق عمل ارقام للعناصر ال اقل منه للمكان ثم بضيف العنصر (راجع الود في الكتاب او اسلايد)

⇒ Removing

الفكرة اني بعد ما احذف العنصر ارجع اعلي مكانه لفاضل ال ايه عن طريق عمل ارقام للامكان، و آخر مكانه بغيره null وارجع العنصر ال بي خزنه فيه العنصر المراد حذفه (راجع الود في الكتاب او اسلايد)



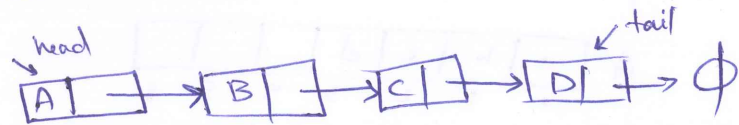
~~Remove~~
Insert



~~insert~~
Remove

3.2 Singly Linked lists (one direction)

- * مجموعة من nodes و node ك element و next reference يوصل من node الى node
- * الانتقال من node الى node بـ next reference
- pointer hopping or link hopping



- * زيارتي في array بـ ترتيب من اليمين الى الشمال
- * لا خدش ما في حجم مبريد ، حيث ان الحجم يتغير حسب ما اريد
- * ما باخر بين الـ index number و node

Insert at Beginning



- * نبي node جديدة ان next يوصل الى head
- * يوصل الى head ان head يوصل الى node الجديدة
- * لازم اطلع ان next يوصل الى node الجديدة يوصل الى head قبل ما اغير

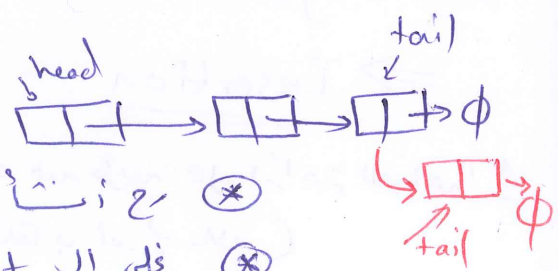
Algorithm addFirst (v):

```

v.setNext(head)
head ← v
size ← size + 1

```

Insert at End



- * مع ان node جديدة ان next يوصل الى null
- * في ان next يوصل الى tail يوصل الى node الجديدة
- * في ان tail يوصل الى node الجديدة
- * لازم ان next يوصل الى tail يوصل الى node الجديدة قبل ما اغير

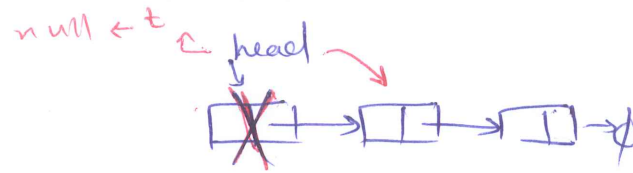
algorithm addLast (v):

```

v.setNext(null)
tail.setNext(v)
tail ← v
size ← size + 1

```

⇒ Removing at Beginning



⊗ رج اخزن ان head في متير

⊗ جلي ان head يترى ان node ان بعد ان يدي اخضر او null اذا ماضى غير

⊗ صلا ان head ان خزنو في متير رج اخضر يا شرع على null (لنود المحذوثة)
(نقد شرع على null)

Algorithm removeFirst():

if head = null then

Indicate an error: the list is empty

t ← head

head ← head.getNext()

t.setNext(null)

size ← size - 1

⇒ Removing at End

الحذف من نهاية ان singly list حسب ويحتاج ان وقت لا نه لا يستطيع الوصول الى آخر
عنصر الا اذا مرت على كل العناصر من بداية ان list وهذا يحتاج وقت ، من
مصلحة نيلجأ ان doubly linked list

R 3.9

method نقرر باضافة عنصر جديد في بداية singly linked list

```
public void insert (String e) {
```

```
    Node n = new Node();
```

```
    n.setElement(e);
```

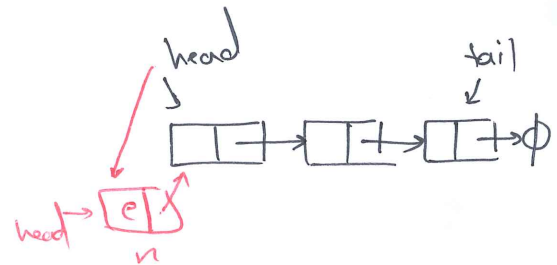
```
    if (size > 0)
```

```
        n.setNext(head);
```

```
    head = n;
```

```
    size ++;
```

```
}
```



R 3.10 singly linked list ۾ Penultimate (پنٽيمائٽ) algorithm

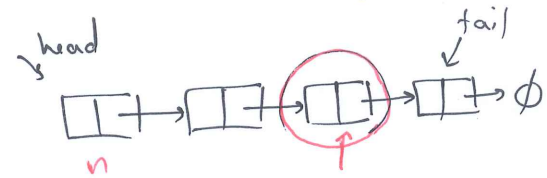
Algorithm findPenultimate():

Node $n \leftarrow \text{head}$

while ($n.\text{getNext}() \neq \text{tail}$) do

$n \leftarrow n.\text{getNext}()$

return n



R 3.8 2 singly linked lists ۾ concatenate algorithm

Algorithm concatenate (L, M)

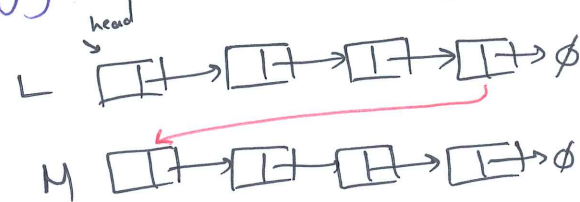
Node $n \leftarrow L.\text{getHead}()$

while ($n.\text{getNext}() \neq \text{null}$) do

$n \leftarrow n.\text{getNext}()$

$n.\text{setNext}(M.\text{getHead}())$

$L \leftarrow L$



Exercise write an algorithm that will calculate the size of a given Linked list L1.

Algorithm calculateSize (L1):

int count $\leftarrow 1$

Node $n \leftarrow \text{head}$

while ($n.\text{getNext}() \neq \text{null}$) do

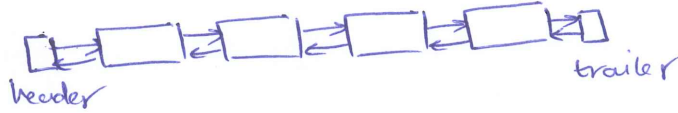
$n \leftarrow n.\text{getNext}()$

count $\leftarrow \text{count} + 1$

return count

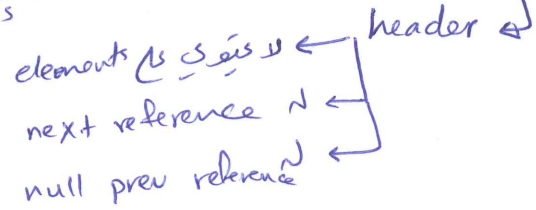
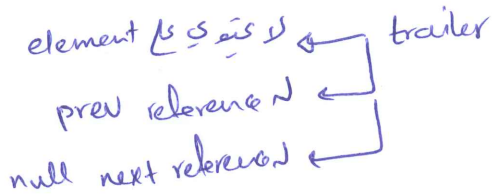
3.3 Doubly Linked Lists (two direction)

① **تتبعين صيغة الترتيب في حبيبتك كما يسجل عملية الحذف والإضافة من البداية والنهاية وحتى منتصفها**

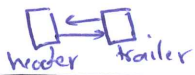


next reference → 2 pointers → ②
prev reference ←

elements لا يتغير في (sentinel nodes or dummy nodes) trailer و header ③



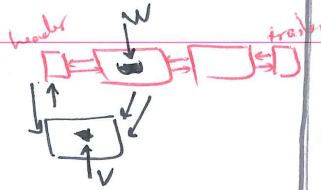
④ **عند إضافة أو حذف عن list لا يجب تغيير header و trailer**
⑤ **Empty list**



Insert

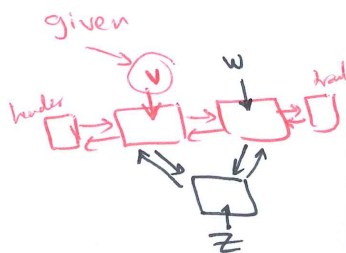
⑥ **at Beginning**

Algorithm addFirst(v):
w ← header.getNext()
v.setNext(w)
v.setPrev(header)
w.setPrev(v)
header.setNext(v)
size ← size + 1



⑦ **in Middle**

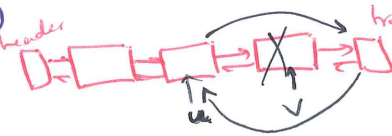
Algorithm addAfter(v, z):
w ← v.getNext()
z.setPrev(v)
z.setNext(w)
w.setPrev(z)
v.setNext(z)
size ← size + 1



Remove

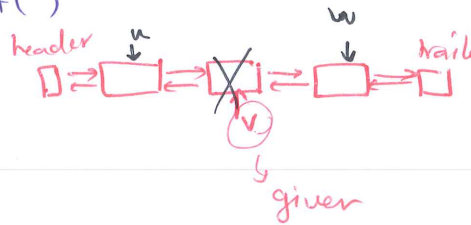
⑧ **at End**

Algorithm removeLast():
if size = 0 then
Indicate an error: the list is empty
v ← trailer.getPrev()
u ← v.getPrev()
trailer.setPrev(u)
u.setNext(trailer)
v.setPrev(null)
v.setNext(null)
size ← size - 1



⑨ **in Middle**

Algorithm remove(v):
u ← v.getPrev()
w ← v.getNext()
w.setPrev(u)
u.setNext(w)
v.setPrev(null)
v.setNext(null)
size ← size - 1



C 3.9 algorithm for joining 2 doubly linked lists L and M into a single doubly linked list L.
 L ← 2 doubly linked lists M

Algorithm concatenateDDL (L, M):

DNode v ← (L.getTrailer()).getPrev()

DNode x ← (M.getHeader()).getNext()

(M.getHeader()).setNext(null)

(L.getTrailer()).setPrev(null)

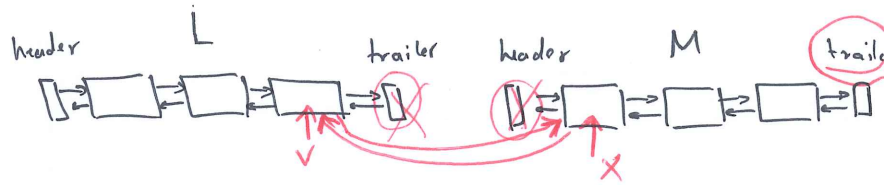
v.setNext(x)

x.setPrev(v)

L̂ = L

L̂.setTrailer(M.getTrailer())

return L̂



C 3.10

singly doubly list if it is a doubly linked list

x y 2 nodes swap algorithm
 y, x is reference in list

⇒ For Singly Linked List

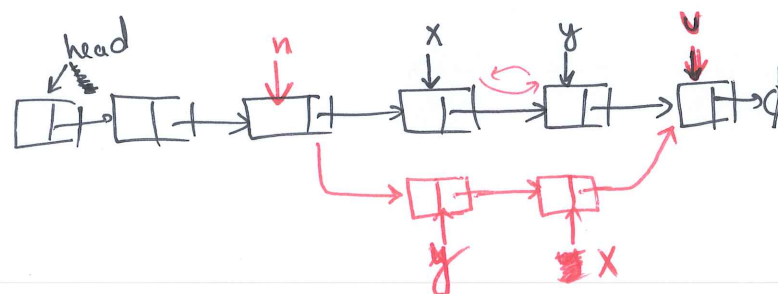
Algorithm swap(x, y):

Node n ← head

while (n.getNext() != x) do {
 n ← n.getNext()

Node v ← y.getNext() → nodes of y

n.setNext(y)
 y.setNext(x)
 x.setNext(v)



⇒ For doubly linked list

Algorithm swapDoubly(x, y):

DNode n ← x.getPrev()

DNode v ← y.getNext()

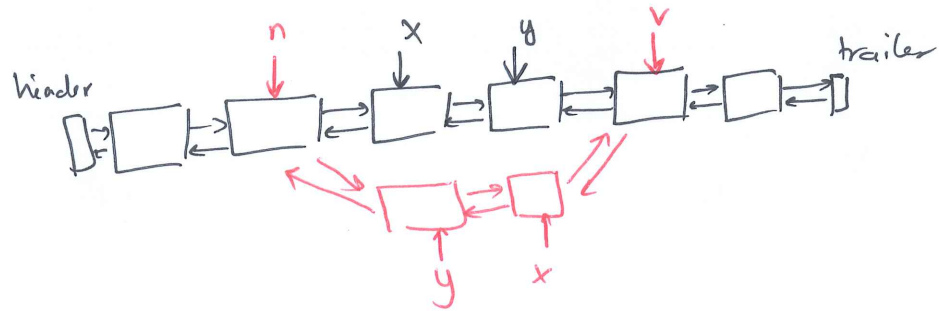
n.setNext(y)

y.setPrev(n)

y.setNext(x)

x.setPrev(v)

v.setPrev(x)



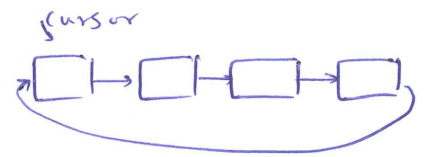
[3.4] Circularly Linked Lists

* no head & no tail

* Last node points to first node

* no first node & no last node

* cursor → traversal
میں سے آگے بڑھنے کے لیے cursor استعمال کیا جاتا ہے اور اس سے آگے بڑھنے کے لیے cursor استعمال کیا جاتا ہے



* Insert after cursor

public void add(Node newNode){

if (cursor == null) {

newNode.setNext(newNode);

cursor = newNode;

} else {

newNode.setNext(cursor.getNext());

cursor.setNext(newNode);

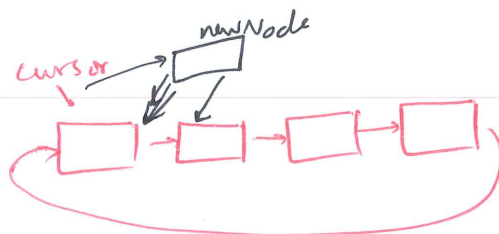
}

size ++;

}

اگر cursor null ہے تو نیا node اس کے بعد
نہیں ہے اور اس کے بعد نیا node ہے

اگر cursor != null ہے تو نیا node اس کے بعد
نہیں ہے اور اس کے بعد نیا node ہے



⊗ Remove after cursor

```
public Node remove() {
```

Node oldNode = cursor.getNext(); → ال node پر پی افسر

if (oldNode == cursor)

cursor = null)

إذا node هاد صا $\leftarrow \text{cursor}$ الى list مع
 خاصية $\text{null} = \text{cursor}$ مع

else {

- `cursor.setNext(oldNode.getNext())`

```
oldNode.setNext(null);
```

3

size ~~to~~ ---;

```
return oldNode;
```

3

```
public void advance () {
```

move the cursor forward,

```
cursor = cursor.getNext();
```

3

R 3.16

Circularly list - Algoric method

```
public int count(L) {
```

```
int counter = 1 ;
```

```
Node n = cursor.getNext();
```

```
while ( n != cursor ) {
```

$n = n \cdot getNext(c);$

Counter ++;

3

```
return counter;
```

3