CS469 Assignment 3

Tianye Zhao, Wenda Yang

March 31, 2019

1

1.1

LinkedList(T info):
  Create a new node pointer $newNode$
  $newNode.value \leftarrow$ info
  $newNode.next \leftarrow$ NULL
  $head \leftarrow newNode$

1.2

$\sim$LinkedList():
  $nodePtr \leftarrow head$
  while $nodePtr \neq$ NULL **do**
    $nextNode \leftarrow nodePtr.next$
    delete $nodePtr$
    $nodePtr \leftarrow nextNode$
  end while

1.3

void appendNode(T info):
  Create a new node pointer $newNode$
  $newNode.value \leftarrow$ info
  $newNode.next \leftarrow$ NULL
  if $head =$ NULL **then**
    $head \leftarrow newNode$
  else
    $nodePtr \leftarrow head$
    while $nodePtr.next \neq$ NULL **do**
      $nodePtr \leftarrow nodePtr.next$
    end while
    $nodePtr.next \leftarrow newNode$
  end if

1.4

T& top():
  return $head.value$

1.5

T& pop_front():
  $tmpNode \leftarrow head$
  $tmpVal \leftarrow head.value$
  $head \leftarrow head.next$
  delete $tmpNode$
  return $tmpVal$

1.6

bool empty():
  if $head =$NULL **then**
    return true
  else
    return false
  end if

1.7

void insertNode(T info):
  Create a new node pointer $newNode$
  $newNode.value \leftarrow$ info
  $newNode.next \leftarrow head.next$
  $head \leftarrow newNode$

1.8

void deleteNode(T info, bool removeAll):
  if $head =$ NULL **then**
    return
  end if
  if $head =$ info **then**
    Create a new node pointer $newNode$
    $newNode \leftarrow head.next$
    delete $head$
    $head \leftarrow newNode$
    if removeAll $=$ **false then**
      return
    end if
  end if
  $nodePtr \leftarrow head$
  while $nodePtr \neq$ NULL **do**

```
    while nodePtr.value ≠ info do
      prevNode ← nodePtr
      nodePtr ← nodePtr.next
    end while
    prevNode.next ← nodePtr.next
    delete nodePtr
    if removeAll = false then
      return
    end if
  end while
```

1.9

```
void displayList(void):
  nodePtr ← head
  while nodePtr ≠ NULL do
    print nodePtr.value
    nodePtr ← nodePtr.next
  end while
```

1.10

```
int count(T val):
  count ← 0
  nodePtr ← head
  while nodePtr ≠ NULL do
    if nodePtr.value = val then
      count ← count + 1
    end if
    nodePtr ← nodePtr.next
  end while
  return count
```

1.11

```
int length():
  len ← 0
  nodePtr ← head
  while nodePtr ≠ NULL do
    len ← count + 1
    nodePtr ← nodePtr.next
  end while
  return len
```

1.12

```
ListNode *getNode(int i):
  count ← 0
```

$nodePtr \leftarrow head$
while $nodePtr \neq$ NULL **do**
if $count =$ i **then**
return $nodePtr$
end if
$nodePtr \leftarrow nodePtr.next$
end while

1.13

T& get(int i):

$count \leftarrow 0$
$nodePtr \leftarrow head$
while $nodePtr \neq$ NULL **do**
if $count =$ i **then**
return $nodePtr.value$
end if
$nodePtr \leftarrow nodePtr.next$
end while

1.14

void clear():

$nodePtr \leftarrow head$
while $nodePtr \neq$ NULL **do**
$nextNode \leftarrow nodePtr.next$
delete $nodePtr$
$nodePtr \leftarrow nextNode$
end while

1.15

void sortBySelection():

$minPtr \leftarrow head$
$nodePtr \leftarrow head.next$
$prevNode \leftarrow head$
while $nodePtr \neq$ NULL **do**
if $nodePtr.value < minPtr.value$ **then**
$minPtr \leftarrow nodePtr$
$minPrevNode \leftarrow prevNode$
end if
$prevNode \leftarrow nodePtr$
$nodePtr \leftarrow nodePtr.next$
end while
if $minPtr \neq head$ **then**
$head \leftarrow minPtr$
$minPrevNode.next \leftarrow minPtr.next$
$minPtr.next \leftarrow head.next$

end if
$curPtr \leftarrow head$
while $curPtr.next \neq$ NULL **do**
$minPtr \leftarrow curPtr.next$
$minPrevNode \leftarrow curPtr$
$nodePtr \leftarrow minPtr.next$
$prevNode \leftarrow minPtr$
while $nodePtr \neq$ NULL **do**
if $nodePtr.value < minPtr.value$ **then**
$minPtr \leftarrow nodePtr$
$minPrevNode \leftarrow prevNode$
end if
$prevNode \leftarrow nodePtr$
$nodePtr \leftarrow nodePtr.next$
end while
if $minPtr = curPtr.next$ **then**
$curPtr \leftarrow curPtr.next$
else
$minPrevNode.next \leftarrow minPtr.next$
$minPtr.next \leftarrow curPtr.next$
$curPtr.next \leftarrow minPtr$
$curPtr \leftarrow minPtr$
end if
end while

1.16

void sortByInsertion():
$endPtr \leftarrow head$
while $endPtr.next \neq$ NULL **do**
$curPtr \leftarrow endPtr.next$
$nodePtr \leftarrow head$
while $nodePtr \neq curPtr$ **and** $nodePtr.value < curPtr.value$ **do**
$prevNode \leftarrow nodePtr$
$nodePtr \leftarrow nodePtr.next$
end while
if $nodePtr = head$ **then**
$endPtr.next \leftarrow curPtr.next$
$curPtr.next \leftarrow head$
$head \leftarrow curPtr$
else if $nodePtr = curPtr$ **then**
$endPtr \leftarrow endPtr.next$
else
$endPtr.next \leftarrow curPtr.next$
$curPtr.next \leftarrow nodePtr$
$prevNode.next \leftarrow curPtr$
end if
end while

1.17

void reverse():

$prevNode \leftarrow head$
$nodePtr \leftarrow head.next$
$nextNode \leftarrow nodePtr.next$
while $nextNode \neq$ NULL **do**
$nodePtr.next \leftarrow prevNode$
$prevNode \leftarrow nodePtr$
$nodePtr \leftarrow nextNode$
$nextNode \leftarrow nextNode.next$
end while
$nodePtr.next \leftarrow prevNode$
$head \leftarrow nodePtr$