# Data Structures 2018
## Exercise 4, solutions (Week 40)

1.-2. See Java files DynArrStack.java and DynArrStackTest.java.

3. There are multiple solutions, for example

   a) $1\,3\,+\,5\,+\,7\,-$

   b) $6\,3\,-\,2\,*\,1\,+$

   c) $3\,4\,3\,*\,2\,+\,*$

   d) $3\,4\,+\,20\,3\,4\,*\,2\,+\,-\,*$

4. Algorithm 1. Complexity is $\mathcal{O}(n)$ because the while-loop is executed $n-1$ times, where $n$ is the lenth of the list.

**Algorithm 1** Find the second to the last node of list $L$

```
SecondLast($L$)
  if $L$.first = null or $L$.first.next = null then
    error: the list $L$ does not contain second to last element
  end if
  $n \leftarrow L$.first
  while $n$.next.next $\neq$ null do
    $n \leftarrow n.next$
  end while
  return $n$
```

5. Algorithm 2. Complexity is $\mathcal{O}(n+m)$, where $n$ is the length of list $A$ and $m$ is the length of list $B$. See 3 for a recursive version.

**Algorithm 2** Merging of lists (a non-recursive version). Variables $h$ and $t$ point to the beginning and the end of the result list $R$, respectively.

```
Merge($A$, $B$)
  if $A$.head = null then
    $h \leftarrow B$.head
    $B$.head ← null
  else if $B$.head = null then
    $h \leftarrow A$.head
    $A$.head ← null
  else
    if $A$.head $<$ $B$.head then
      $h \leftarrow A$.head
      $A$.head ← $A$.head.next
    else
      $h \leftarrow B$.head
      $B$.head ← $B$.head.next
    end if
    $t \leftarrow h$
    while $A$.head $\neq$ null and $B$.head $\neq$ null do
      if $A$.head $<$ $B$.head then
        $t$.next ← $A$.head
        $A$.head ← $A$.head.next
      else
        $t$.next ← $B$.head
        $B$.head ← $B$.head.next
      end if
      $t \leftarrow t$.next
    end while
    if $A$.head = null then
      $t$.next ← $B$.head
      $B$.head ← null
    else
      $t$.next ← $A$.head
      $A$.head ← null
    end if
  end if
  $R \leftarrow$ new list
  $R$.head ← $h$
  return $R$
```

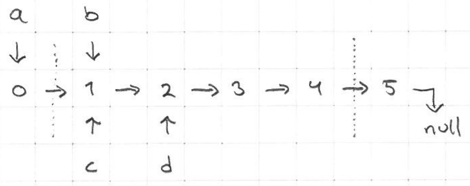
**Algorithm 3** Merging of lists (a recursive version).

```
MergeRec($a$, $b$)
  if $a$ = null then
    return $b$
  else if $b$ = null then
    return $a$
  else if $a < b$ then
    $a$.next ← MergeRec($a$.next, $b$)
    return $a$
  else
    $b$.next ← MergeRec($a$, $b$.next)
    return $b$
  end if
Merge($A$, $B$)
  $R$ ← new list
  $R$.head ← MergeRec($A$.head, $B$.head)
  $A$.head ← null
  $B$.head ← null
  return $R$
```

6. Variables $a$ and $b$ contain $x-1$th (if $x > 0$) and $x$th elements of the list. In the second for loop, the next pointer of element $L[i+1]$ (variable $d$) is changed so that it points to element $L[i]$ (variable $c$). Finally, $c$ contains element $L[y]$, $d$ contains element $L[y+1]$ or null, if $y = n-1$. If $y = n-1$, we have to iterate through the whole list. Hence, the time complexity of the algorithm is $\mathcal{O}(n)$. The functioning of the algorithms is visualised in the accompanying figure.

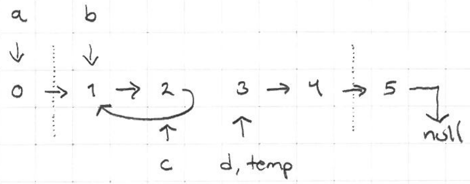
L = [0, 1, 2, 3, 4, 5], x = 1, y = 4

Situation after the first for loop and initialisation of pointers c and d :

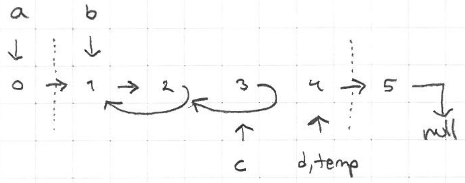


Situation after each iteration of the second for loop :

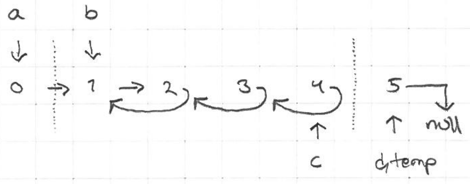
i = 1



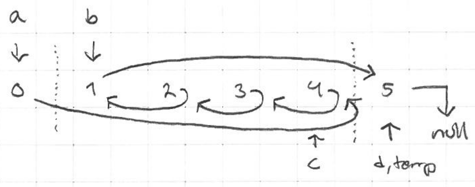
i = 2



i = 3



Situation after line "b.next ← d" and final conditional statements :



```
ReversePart($L, x, y$)
  if $L$.head = null then
    error : the list is empty
  end if
  $a \leftarrow$ null
  $b \leftarrow L$.head
  for $i \leftarrow 0$ to $x - 1$ do
    $a \leftarrow b$
    $b \leftarrow b$.next
    if $b$ = null then
      error : index is out of bounds
    end if
  end for
  $c \leftarrow b$
  $d \leftarrow c$.next
  for $i \leftarrow x$ to $y - 1$ do
    if $d$ = null then
      error : index is out of bounds
    end if
    $temp \leftarrow d$.next
    $d$.next $\leftarrow c$
    $c \leftarrow d$
    $d \leftarrow temp$
  end for
  $b$.next $\leftarrow d$
  if $x = 0$ then
    $L$.head $\leftarrow c$
  else
    $a$.next $\leftarrow c$
  end if
  return $L$
```

7. The idea: The digits of the numbers $A$ and $B$ are compared pairwise from right to left (The digits are stored in this order to the lists). In this procedure we store the result of the latest digit comparement to the variable $diff$. If the integers have different number of digits the shorter list is filled with zeros to make the list lengths equal (For example integers 125 and 23 are compared as 125 and 023). If number is negative then the last element of the list is -1.

8. (a) i) A
       ii) E, I, G, H, D
       iii) A, B, C, F
       iv) A
   (b) i) E, F, G
       ii) A, B
       iii) B, E, F, G, I
       iv) C, D

```
ReadNumber($x$)
if $x = \texttt{null}$ then
    $num \leftarrow 0$
else if $x$.val $=$ '$-$' then
    $num \leftarrow -1$
else
    $num \leftarrow x$.val
end if
return $num$
```

```
Next($x$)
$n \leftarrow x$
if $n \neq \texttt{null}$ then
    $n \leftarrow n$.next
end if
return $n$
```

```
IsGreater($A$, $B$)
precondition: $A$ and $B$ contain correct list representations of an integer
$a \leftarrow A$.head
$b \leftarrow B$.head
$diff \leftarrow 0$
while $a \neq \texttt{null}$ or $b \neq \texttt{null}$ do
    $aNum \leftarrow$ ReadNumber($a$)
    $bNum \leftarrow$ ReadNumber($b$)
    if $aNum > bNum$ then
        $diff \leftarrow 1$
    else if $aNum < bNum$ then
        $diff \leftarrow -1$
    else if $aNum = bNum = -1$ then
        $diff \leftarrow -diff$
    end if
    $a \leftarrow$ Next($a$)
    $b \leftarrow$ Next($b$)
end while
if $diff = 1$ then
    return true
else
    return false
end if
```