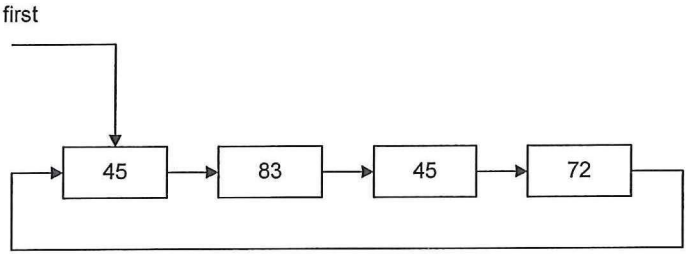
Tentamen TI1310 Algoritmen en Datastructuren, 20 april 2012, 9.00-12.00
TU Delft, Faculteit EWI, Basiseenheid Software Engineering
Open boek tentamen, alleen het boek van Goodrich en Tamassia is toegestaan.

Opgave 1 (30 punten)

In Bijlage 1 is de specificatie gegeven van klasse Node. Een Node-object bevat een waarde en een verwijzing naar een andere Node. Bestudeer de specificatie grondig voordat je begint aan de rest van deze opgave

Met behulp van klasse Node wordt een klasse Queue geïmplementeerd. Deze klasse heeft methoden voor het toevoegen (add) en verwijderen (remove) van waarden. Verder zijn er de methoden getSize, toString en reverse. Methode reverse keert de inhoud van de queue om. Bestaat de queue bijv. uit `[1,6]` dan is de inhoud na uitvoering van reverse `[6,1]`.

De queue wordt geïmplementeerd als een circulaire lijst van Nodes. Er is een attribuut first dat het 1e element uit de circulaire lijst aanwijst. Zie de tekening hieronder. Toevoegen gebeurt aan de achterkant van de lijst (dus na de knoop met 72), verwijderen gebeurt aan de voorkant van de lijst (dus bij first).



Gegeven de specificatie van klasse Queue:

```
public class Queue{
      private Node first;
      private int size;

      public Queue()
      post: has constructed an empty Queue

      public void add(int v)
      post: has added a Node containing v to the rear of queue

      public int remove()
      pre:  the queue is not empty
      post: if pre is true, has removed the first Node and returned the value of
      this  Node, has otherwise thrown a Runtime exception.

      public int size()
      post: returns the size of the queue

      public void reverse()
      post: has reversed the order of the values of the queue

      public String toString()
      post: returns String representation of the Queue
}
```

Vragen:

a. Geef een implementatie van methode add().
b. Geef een implementatie van methode remove().
c. Geef een implementatie van methode reverse().

d. Wat is de tijdcomplexiteit van methode reverse()? Licht je antwoord toe.
e. Wat is de tijdcomplexiteit van methode toString()? Licht je antwoord toe.

Opgave 2(15 punten)

Voor het doorlopen van een Queue wordt een QueueIterator ontwikkeld. De iterator levert de waarden geordend op grootte, van **klein naar groot**. Van waarden die 2 of meer keer voorkomen, **wordt maar 1 exemplaar opgeleverd.** Bij de queue die in het voorbeeld is getekend, zijn dat de (3!) waarden: 45, 72 en 83.

```
public class QueueIterator{

      //declaratie van attributen van de iterator

      public QueueIterator(Queue q)
      post: has constructed an new QueueIterator

      public void reset()
      post: has reset the iterator to the beginning of the traversal

      public Object next()
      pre:  hasNext() is true
      post: returns if pre is the current value and increments iterator

      public boolean hasNext()
      post: returns true iff not all elements has been visited
}
```

Vragen:

a. Geef een declaratie van de attributen van QueueIterator. Leg uit wat de rol is van elk van de attributen. Gebruik (indien nodig) een of meer datastructuren uit het boek.

b. Geef een implementatie van de constructor.
c. Geef een implementatie van methode reset().
d. Geef een implementatie van methode next().
e. Geef een implementatie van methode hasNext().

f. Wat is de tijdcomplexiteit van methode reset()? Licht je antwoord toe.
g. W at is de tijdcomplexiteit van methode next()? Licht je antwoord toe.

Opgave 3(15 punten)

Gegeven een recursieve methode schrijfMiddens, die de elementen van een ArrayList afdrukt. Bij het afdrukken van de elementen wordt eerst het middelste element van de ArrayList afgedrukt, daarna de linkerhelft en dan de rechterhelft. Bij het afdrukken van de linkerhelft, wordt eerst het middelste element van de linkerhelft afgedrukt, daarna de linkerhelft van de linkerhelft en daarna de rechterhelft van de linkerhelft, enz. Voorbeeld: de ArrayList met inhoud `[0,1,2,3,4,5,6]` wordt afgedrukt in de volgorde `3,1,0,2,5,4,6.`

```
public static void schrijfMiddens(int og, int bg, ArrayList<Integer> aList){
      if (og <= bg){
            int mid = (og + bg)/2;
            System.out.print(aList.get(mid) + " ");
            schrijfMiddens(og,mid-1,aList);
            schrijfMiddens(mid+1,bg,aList);
      }
}
```

De methode aangeroepen als `schrijfMiddens(0,aList.size()-1,aList);`

a. De differentievergelijking voor het aantal aanroepen van schrijfMiddens is (bij benadering) gelijk aan f(n) = 2f(n/2), met f(1) = 1. Leid een oplossing voor de een differentievergelijking af. Laat zien hoe je dat doet.

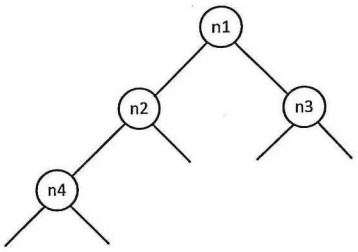
Een recursieve methode kan worden herschreven als een iteratieve methode, die gebruik maakt van een stack.

b. Herschrijf schrijfMiddens als een iteratieve methode. Maak gebruik van een Stack-implementatie uit het boek.

c. Stel dat in de methode die je bij 3b hebt geschreven de stack wordt vervangen door een queue, in welke volgorde zou dan de inhoud van de ArrayList [0,1,2,3,4,5,6] worden afgedrukt. Licht je antwoord toe.

Opgave 4 (30 punten)

In Bijlage 2 vindt je de implementatie van de klasse BinNode. De klasse implementeert knopen waarmee een binaire boom kan worden gevormd. Elke knoop heeft drie attributen: value die een geheel getal bevat en left en right die verwijzen naar de linker- en rechter sub-boom. Hiernaast is een binaire boom getekend. De boom bevat 4 knopen, waarvan er twee (n3 en n4) een blad zijn. **Een blad is een knoop met twee lege sub-bomen.**



Gevraagd wordt om de implementatie van een aantal statische methoden voor een binaire boom. Geef als je gebruik maakt van hulpmethoden, de pre- en postconditie en de volledige implementatie van de hulpmethoden.

a. Geef een implementatie van een methode isLeaf(BinNode n) die bepaalt of een knoop een blad is.

```
public static boolean isLeaf(BinNode n)
post: returns true iff n is a leaf
```

b. Geef een implementatie van een methode numberOfleaves(BinNode n) die het aantal bladeren in een binaire boom berekent.

```
public static int numberOfLeaves(BinNode n)
post:returns the number of leaves in the binary tree with root n
```

c. Geef een implementatie van een methode leavesPerLevel(BinNode n) die een queue geeft met het aantal bladeren per niveau van de boom.

```
public static Queue leavesPerLevel(BinNode n)
post: returns a queue that contains the number of leaves at each
level, of the tree with root n
```

Toelichting: In de tekening is het aantal bladeren op het hoogste niveau 0, op het volgende niveau 1 en op het daaropvolgende niveau 1. De queue voor deze boom is: [0,1,1].

Bijlage 1

```
public class Node{

      private int value;
      private Node next;

      public Node(int v, Node n){
            setValue(v);
            setNext(n);
      }

      public int getValue(){          return value;      }
      public Node getNext(){          return next;       }
      public void setValue(int v){    value = v;         }
      public void setNext(Node n){    next  = n;         }
      public String toString(){       return "<Node("+value+")>";    }
}
```

Bijlage 2

```
public class BinNode{

      private int value;
      private BinNode left;
      private BinNode right;

      public BinNode(int v, BinNode l, BinNode r){
            setValue(v);
            setLeft(l);
            setRight(r);
      }

      public int getValue(){                 return value;      }
      public BinNode getLeft(){              return left;       }
      public BinNode getRight(){             return right;      }
      public void setValue(int i){           value = v;         }
      public void setLeft(BinNode l){        left  = l;         }
      public void setRight(BinNode r){       right = r;         }
}
```