Exam – extra part: solution

Tasks: solutions

Task 1 (4 points + 3 points)

a) (4 points)

```
public static void sort (int[] numbers)
{
    int    lastPos = numbers.length - 1;
    for (int pos = lastPos; pos > 0; pos--)
    {
        for (int p = 0; p < pos; p++)
        {
            if (numbers[p] > numbers[p + 1])
            {
                int    e = numbers[p];
                numbers[p] = numbers[p + 1];
                numbers[p + 1] = e;
            }
        }
    }
}
```

b) (3 points)

The worst case appears when the integer sequence is already sorted in reverse. In this case there will be $n - 1$ element exchanges in the first pass of the main loop, $n - 2$ element exchanges in the second pass, and so on. In the last pass through the main loop there will be 1 exchange. The total number of exchanges is:

$$(n - 1) + (n - 2) + ... + 2 + 1$$

This means that the worst case time complexity of the algorithm, in terms of element exchanges, can be given by the following complexity function:

$$w(n) = n\,(n - 1) / 2$$

This function can also be written like this:

$$w(n) = n^2/2 - n/2$$

The term n^2 dominates for sufficiently large values of n, and therefore:

$$w(n) \in \Theta(n^2)$$

In the worst case the algorithm is quadratic, in terms of element exchanges.

Task 2 (3 points)

Definition: the set $O(n \log_2 n)$

A complexity function $f(n)$ belongs to the set $O(n \log_2 n)$ if and only if there exists a real, positive constant c and a non-negative integer N, for which the following inequality holds for all $n \geq N$:

$$f(n) \leq c\, n \log_2 n$$

Task 3 (4 points + 3 points + 3 points)

a) (4 points)

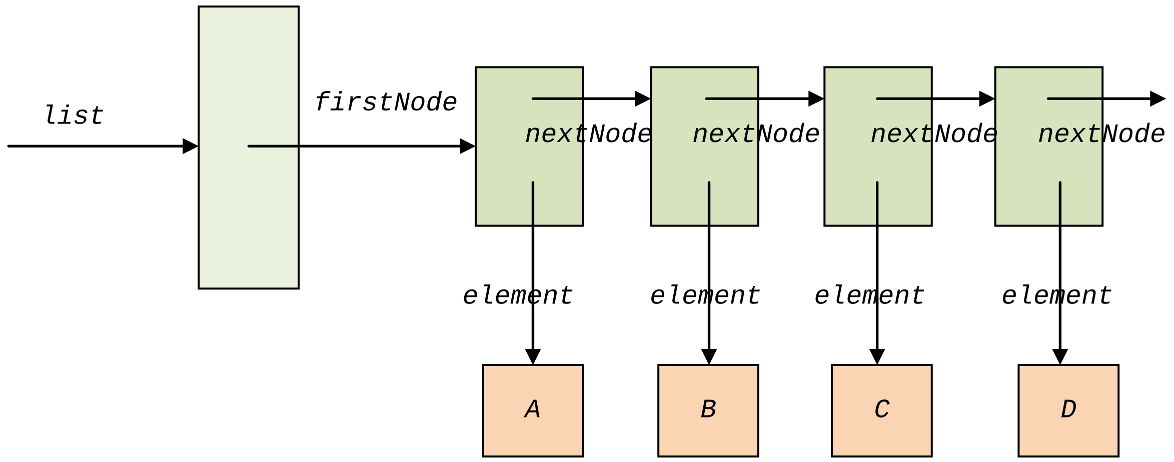
```
// add appends a given element to the end of the list
public void add (E element)
{
    Node    newNode = new Node (element);

    if (firstNode == null)
        firstNode = newNode;
    else
    {
        Node    node = firstNode;
        while (node.nextNode != null)
            node = node.nextNode;
        node.nextNode = newNode;
    }
}
```

b) (3 points)

```
// get returns the element at a given position
public E get (int index)
{
    Node    node = firstNode;
    for (int i = 0; i < index; i++)
        node = node.nextNode;
    return  node.element;
}
```

c) (3 points)


list
firstNode
nextNode
nextNode
nextNode
nextNode
element
element
element
element
A
B
C
D