Outline of the proposed solutions for selected problems.

1. (5 p)
   (a) The justification is not included. See similar problems in lektionerna.
      - false
      - true
      - false
      - true
   (b) The worst case is when the numbers are relatively prime, in which case $gcd(m, n) = 1$. This means that the algorithm needs to check $\min(m, n)$ numbers. Thus the worst-case complexity is $\Theta(\min(m, n))$.
      - Note that $m > n$ at each step of the algorithm, possibly with except of the initial state. The lemma shows that if $m > n$ then after each two consecutive steps of the algorithm the value of $n$ is reduced at least by half. Thus the algorithm runs in time $\Theta(2 \log(\max(m, n)))$, thus in $\Theta(\log n)$. Proof of the lemma is not included.

2. The answer is based on [G&T] 4th edition p.208. (3 p)
   (a) Minimal size of the ring buffer is 4. This is because for the reasons discussed in p.209 the queue can never hold more than $N - 1$ objects and while performing the given sequence maximal number of objects on the queue is 3.
      The output values appear in the order of their enqueueing, i.e.: 5, 3, 2, 8, 9
      The last number (1) remains in the queue, as there are 6 enqueue operations and only 5 dequeue.
      The final state depends on the size of the buffer. For buffer of size 4 (indexed 0..3) implemented as expained in p.208 of the course book we get:
      $f = 1$, $r = 2$, Buffer = [9,1,2,8]
   (b) Create an empty stack $S$. Given postfix expression $E$ is read from left to right, item by item. If a consecutive item is a number it is placed on the stack. Otherwise, it is an operation $op$. In that case the stack is not empty, and its two topmost elements are numbers. We execute then the sequence of operations: $x \leftarrow pop(S), push(op(x, pop(S)), S)$. After processing of the last item of the expression the stack includes only one number which is the value of the expression $E$. 
The complexity of the algorithm is linear on the number of items of $E$ since its items are processed only once and processing of an item consists in at most three stack operations and one arithmetic operation.

- The contents of the stack on the consecutive steps of the evaluation of the example expression is as follows.

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

3. (3 p)

(a) 

\[ [9, 4, 15, 8, 5] \]
\[ [4, 9, 15, 8, 5] \]
\[ [4, 9, 15, 8, 5] \]
\[ [4, 8, 9, 15, 5] \]
\[ [4, 5, 8, 9, 15] \]

(b) 

\[ \text{aba, dab, bbc, ccc, abd, cad} \]
\[ \text{dab, cad, aba, bbc, abd, ccc} \]
\[ \text{aba, abd, bbc, cad, ccc, dab} \]

(c) As the sequence $S$ has 7 elements, to compute median we call \texttt{quickSelect(S, 4)} [G&T] p.530.

Assuming that the first element is selected as pivot we get the following division: $L = [3, 2]$, $E = [5]$, $G = [8, 7, 9, 12]$. and the recursive call \texttt{quickSelect(G, 1)}. The division gives $[7], [8], [9, 12]$ and the last recursive call is \texttt{quickSelect([7], 1)} immediately returning 7 as the result.

4. (3 p)

(a) 

The tree

```
  41(-1)
 /   \
16(-1) 67(1)
  \
 39 49(0) 74
  \
 47 61
```

is an AVL tree, since the balance of each node is 0, 1 or -1.
Insert(40) final result:

\[ T_1 = \begin{array}{c}
41 \\
/ \\
39 67 \\
/ \\
16 40 49 74 \\
/ \\
47 61 
\end{array} \]

Insert(53) The critical node is 67. Insertion will include 53 as the left child of 61. After that re-balancing is needed. It is achieved by two rotations: left rotation around 49:

\[ T_2 = \begin{array}{c}
41 \\
/ \\
16 67 \\
/ \\
39 61 74 \\
/ \\
49 \\
/ \\
47 53 
\end{array} \]

and right rotation around 67. Final result:

\[ T_3 = \begin{array}{c}
41 \\
/ \\
16 61 \\
/ \\
39 49 67 \\
/ \\
47 53 74 
\end{array} \]

Delete(67) First step replaces 67 by its inorder successor 74:

\[ T_4 = \begin{array}{c}
41 (-1) \\
/ \\
(-1)16 74 (2) \\
/ \\
39 49 (0) \\
/ \\
47 61 
\end{array} \]

This is not an AVL tree; the balance of 74 is to be restored by right rotation. The obtained tree is an AVL tree.

\[ T_5 = \begin{array}{c}
41 (-1) \\
/ \\
(-1)16 49 (-1) \\
/ \\
39 47 74 (1) \\
/ \\
61 
\end{array} \]
The above solution shows application of each the operations to the original tree. If each of the operation is applied to the result of the previous one then: (1) T2' obtained by insertion of 53 to T1 has the left subtree as in T1 and right subtree as in T2; (2) Removal of 67 from T2' does not destroy the AVL property and does not require any re-balancing.

(b) Heaps: Tables are omitted.

```
  6
   /  
  5   2
  / \  /  
 6  5 4  5
```

```
  2
 / \ 
 4  5  
 /   
 6
```

```
  2
 / \ 
 4  5  
 /   
 6  7
```

```
  1
 / \ 
 4  2
 / \  /  
 6  7  5
```

```
  1
 / \  
 4  2  
 / \   
 6  7  5  3
```