1. The application of abstract data types:
   (a) Demands much discipline even in a programming language such as Java
   (b) Is possible only when a programming language such as Java is used
   (c) Happens automatically when you use the programming language Java

2. The relation between *abstract data types* and (*Java*) *interfaces* is:
   (a) They are two different names for the same concept
   (b) The concept ‘interface’ is an extension of the concept ‘abstract data type’
   (c) Interfaces are suitable to express abstract data types in Java

3. During the determination of the ‘Big-Oh’ notation for the *running time* T(n) of an algorithm, we can neglect the actual time spent by an elementary operation, because:
   (a) We are only interested in an approximation
   (b) It does not influence the asymptotic behavior of the algorithm
   (c) The actual time spent cannot be measured

4. Which running time indicates a *polynomial* algorithm?
   (a) $T(n) = O(n \log(n))$
   (b) $T(n) = O(n^2 + n^3)$
   (c) $T(n) = O(2^n)$
   (d) $T(n) = O(\log(n))$

5.What is the asymptotic *worst-case* running time (in Big-Oh notation) of the following variant of Bubblesort algorithm:

```
void sort(String A[], int n) {
 boolean bubbled = false;
 for (int i=1; i<n; i++)  {
   bubbled = false;
   for (int j=0; j<n-i; j++) {
     if (A[j] >= A[j+1]) {
        String y = A[j]; A[j] = A[j+1]; A[j+1]= y;
        bubbled = true;
     }
   }
   if (!bubbled) break;
 }
}
```

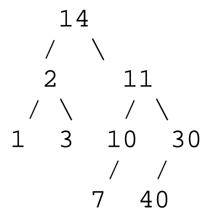
   (a) $T(n) = O(n)$
   (b) $T(n) = O(2n + n \log(n))$
   (c) $T(n) = O(n^2)$
   (d) $T(n) = O(n \log(n))$

6. What is the running time of the algorithm in question 5 in the case that array A[] is already sorted (ascending)?
   (a) $T(n) = O(n)$
   (b) $T(n) = O(2n + n \log(n))$
   (c) $T(n) = O(n^2)$
   (d) $T(n) = O(n \log(n))$

7. Assume, a *Stack* ADT is (efficiently) implemented with a single Queue. What can you tell about the complexity of Push() and Pop()?
   (a) Both operations have to be O(n) (n is the number of elements on the Stack)
   (b) One of them can be O(1) and the other one has to be O(n)
   (c) One of them can be O(n) and the other one has to be $O(n^2)$

8. Observe the concepts *position and rank of an element* in a Sequence based on an Array:
   (a) Both the *rank* and the *position* of an element agree with the index of the element in the Array
   (b) The *rank* of an element agrees with the index of the element in the Array, the *position* is relative and is determined with the aid of the index.
   (c) The *position* of an element agrees with the index of the element in the Array, the *rank* is relative and is determined with the aid of the index.
   (d) Neither the *position* nor the *rank* have any relation with the index of the Array

9. What is the *in-order* list of the elements in de following binary tree?



   (a) 14 – 2 – 1 – 3 – 11 – 10 – 7 – 30 – 40
   (b) 1 – 3 – 2 – 7 – 10 – 40 – 30 – 11 – 14
   (c) 1 – 2 – 3 – 14 – 7 – 10 – 11 – 40 – 30
   (d) 1 – 2 – 3 – 7 – 10 – 11 – 14 – 30 – 40

10. We want to use an Euler-tour to compute an expression (e.g.: 3*5 / 2 + 5) that is represented in a binary tree. The Euler-tour is executed by the *Template-method pattern*. In which of the following four methods do we perform the operations (+, -, /, *, etc.)?
   (a) VisitLeft
   (b) VisitBelow
   (c) VisitRight
   (d) VisitExternal

11. The *Heap-order property* means that:
   (a) The left child of a node always has a larger value than the right child
   (b) All children at the left side of a node have smaller values than the children at de right side of a node
   (c) Children do not have smaller values than their parents
   (d) Nodes at depth d always have larger values than nodes at depth d-1

12. During the insertion of an element in a Binary Search Tree,
   (a) The element is always placed in an external node that becomes internal
   (b) First, room is made in an internal node by moving some elements
   (c) One or more *restructures* are performed after the element is placed

13. AVL trees and Splay-trees both make use of rotations (*tri-node restructure operations*). In AVL trees, this can happen at insertions and deletions, in Splay trees it happens more than once at *all operations*.
   (a) It means that Splay trees are always less preferable than AVL trees because AVL trees need less rotations
   (b) The height of AVL trees is guaranteed to be O(log n); with Splay trees this is not the case. So, AVL trees are to be preferred.
   (c) Splay trees are preferable if some elements are used more often than others
   (d) Splay trees are better balanced than AVL trees and thus are preferable.

14. To sort small numbers of items:
    (a) You have to use a quadratic algorithm
    (b) It is unwise to use an O(n log n) sorting algorithm
    (c) You can better use Insertion-Sort than Quick-Sort
    (d) It does not matter which algorithm is used

15. The cut-off value in Quick-Sort indicates:
    (a) The spot where the list to sort is split in two
    (b) The maximal size of a list to sort on which Quick-Sort still is used
    (c) The minimal size of a list to sort on which Quick-Sort still is used
    (d) The maximal depth of the recursion

16. Dynamic programming is to be preferred over Divide-and-Conquer when:
    (a) There are many subproblems
    (b) Subproblems do overlap much
    (c) The problem cannot be divided into subproblems
    (d) The solution of a problem is not easily obtained from solutions of subproblems

17. The advantage of the Hirschberg algorithm for finding the *Optimal Alignment* of two DNA-strings over the *Longest Common Substring*-method from the handbook is:
    (a) Hirschberg applies Divide-and-Conquer and on dynamic programming
    (b) The running time of the Hirschberg-algorithm is much lower
    (c) (Much) less storage space is needed for the Hirschberg-algorithm
    (d) The algorithms are in fact identical. There is no advantage

18. A *back edge* appears during depth-first search in a directed graph if:
    (a) A node does not have children
    (b) A child of a node already has been visited and also is a descendent of that node
    (c) A child of a node already has been visited but is not a descendent of that node
    (d) An arrow crosses an arrow that is already visited before

19. Finding substring T of string S in a *Suffix Trie* of S costs (not including the construction of the Suffix Trie):
    (a) O(|S|) operations
    (b) O(|T|) operations
    (c) O(|S|+|T|) operations
    (d) O(|S| x |T|) operations

20. In a *region-based* k-D tree:
    (a) Areas are divides in two equal parts on each internal node
    (b) The distribution of points in the area determines the way in which the area is split
    (c) The size of the area determines whether an area is split

Open question:
Describe three different applications of *Tries* in Knowledge Engineering.