Linnæus University
School of Computer Science,
Physics and Mathematics

Examination

Examiner
Dr Jonas Lundberg

Date
2014–10–29

Time
8–13

Place
Stenladan, Stallvägen 10, 2nd floor

Course Code
2DV100

Allowed aids
None.

Messages from the teacher
Exercises: 5
Maximum points: 50 p
Pass: 25 p

Points

Grade

The student complete the form below

Name: __________

Address: __________

E-mail: __________

Telephone: __________

Civic reg. number: __________

The number of sheets handed in: __________

Tick the exercises you hand in

Exercises	**Points**	*Put a tick*
1		
2		
3		
4		
5		

Uppvisat kårlegitimation ☐ Ja ☐ Nej

Uppvisat legitimation ☐ Ja ☐ Nej

Tid för inlämnande

Tentamensvaktens signatur

Linnæus University
School of Computer Science,
Physics and Mathematics
Dr Jonas Lundberg

Examination in Computer Science, 2DV100, 7.5cr
October29, 2014, 8.00–13.00

Allowed aids: None.

1. **(a)** The interface below specifies the functionality of a simple integer queue. A queue is a sequential collection with add and remove at different sides. That is, a *FiFo* (First in, First out) data structure. Write a Java class `Queue.java` that provides an implementation of this interface. You can either do a *linked* implementation or an *array-based* implementation. But you are not allowed to use any of the collection classes in the Java Library.

```
public interface IntQueue {
    int size();                          // Current queue size
    boolean isEmpty();                   // true if queue is empty
    void enqueue(int element);           // Add element at the end of the queue
    int dequeue();                       // Returns (and removes) first element in queue.
                                         // Exception if queue is empty
    String toString();                   // Returns a string representation of queue content
}
```

(10p)

2. **(a)** Write down a static public method `hasSubstring(String str, String sub)` that returns `true` if the string `str` contains the string `sub` as a substring (and `false` if it does not). That is, if we can find `sub` somewhere inside `str`. For example,

```
hasSubstring("Hello World!", "Hel")          ==> true
hasSubstring("Hello World!", "lo W")         ==> true
hasSubstring("Hello World!", "Hej")          ==> false
hasSubstring("Hello World!", "world")        ==> false
```

(5p)

(b) Write down a static public method `sort(int[] arr)` creating and returning a new array, consisting of all elements in `arr` in a sorted order (lowest first). Array `arr` should be left unchanged. **Notice:** You should do the sorting yourself. You are not allowed to use any of the predefined sorting algorithms in the Java Library. (5p)

3. **(a)** What is an *iterator* in Java? Why do we use them? How do we use them? And how are they related to the `Iterable` interface in Java. (4p)

(b) Consider a binary search tree for integers where (as usual) smaller elements are stored in the left subtree. (6p)

1. Sketch the tree that is the result of adding the numbers

   `23, 7, 27, 46, 52, 9, 12, 8, 6, 5`

   to the tree (in given order).

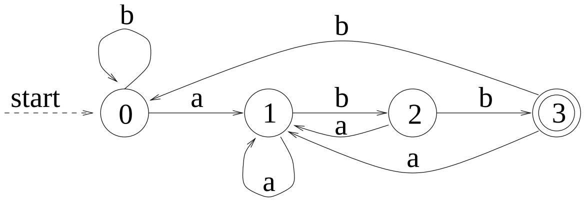
2. What is printed if you apply the following algorithm to the tree above (starting in the root node)?

   1: Print the left subtree (if it exist)
   2: Print the right subtree (if it exist)
   3: Print the node value

3. Write down an algorithm for finding the highest value in a binary search tree.

4. **(a)** Create a deterministic finite automata (DFA) that acepts the language with an *odd* number of as and an *odd* number of bs. Example of acceptable strings are $ab, babb, ababab, bbbaaaaa$. (3 p)

(b) Convert the DFA below to a regular expression using the *State Elimination* technique. Show step-by-step how the regular expression is created. (3 p)



(c) Construct a context-free grammar which generates the regular language $L(R)$ that can be specified by regular expression $R = 0^*1(0^+1)^*$. Also, do a left-most derivation and sketch the corresponding parse tree for the following strings:

1) 00101
2) 1001

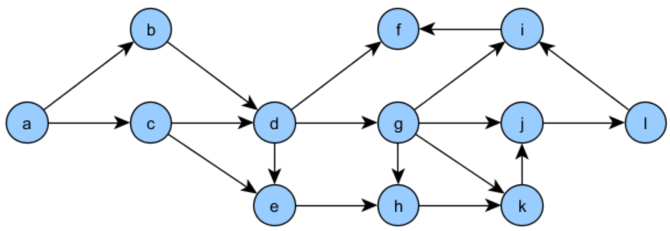
(4 p)

5. **(a)** The table below shows the execution times for four different algorithms, for four sizes of input data (n). The time complexity classes of the algorithms are $O(n^2)$, $O(1)$, $O(n \lg n)$ and $O(n)$. Pair the time complexity classes with the functions. Motivate your answers!

(4p)

n	$T_1(n)$	$T_2(n)$	$T_3(n)$	$T_4(n)$
100	33	30	20	100
200	76	60	80	100
500	224	150	500	100
1000	498	300	2000	100

(b) For the graph below list nodes in: a) postorder(a), b) topological order(a). (3 p)



(c) For the graph below compute the Transitive Closure. Describe the algorithm used. Estimate (and motivate) the time-complexity of the used algorithm. (3p)