OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

2016/2017

Semester 1 - Winter 2016

CS3509: Theory of Computation

Dr. Helen Purchase (Extern)
Professor Cormac Sreenan
Professor Michel Schellekens

One and a half hour

Total marks: 80

Answer all Questions

Calculators allowed

PLEASE DO NOT TURN THIS PAGE UNTIL INSTRUCTED TO DO SO

ENSURE THAT YOU HAVE THE CORRECT EXAM PAPER

Question 1 [20 marks]
a) [14 marks] Demonstrate that you cannot write a program that checks for any given Python program P whether this program will ever output 33.

b) [3 marks] Demonstrate that the set of all finite sequences consisting of non-zero natural numbers is a countable set. You can base your argument on bijective functions or injective functions that we have encountered in the course. For such functions you do not need to prove that they are bijective or injective.

c) [3 marks] We have seen that the maximal independent set problem is polynomial time equivalent to the minimal vertex cover problem. Consider a graph $G = (V, E)$, where V is the set of vertices and E is the set of its edges. Which equivalence has been used to link the maximal independent set S to a minimal vertex cover? You do not need to give the proof. Simply state the equivalence that relates the maximal independent set to the minimal vertex cover, expressed in terms of the set S and the set V.

Question 2 [20 marks]
Provide the decision tree for the Selection Sort algorithm for input-lists of size 3.
We recall the algorithm's pseudo-code.

```
Selection Sort(L)
for $i = 1$ to $|L| - 1$ do
    $k \leftarrow i$
    $l \leftarrow L[i]$
        for $j = i + 1$ to $|L|$ do
            if $L[j] < l$ then $k \leftarrow j$
                               $l \leftarrow L[j]$
    SWAP($L[i], L[k]$)
```

a) [6 marks] On each branch-part of the decision tree (leading from a parent node to its child), indicate the comparison between list elements that led to the creation of this branch.

b) [6 marks] At each node, indicate the input lists for which the comparison path in the tree will lead from the root node to that particular node.

c) [6 marks] At each node, indicate the list recording the swaps carried out along the comparison path leading to that node. We refer to this list as the "swap-results-list."

d) [2 marks] For each leaf, check that the swaps indicated by the swap-results-list do sort the input list recorded at the leaf.

Question 3 [20 marks]
Is it possible for a comparison-based sorting algorithm to sort 2^n of its inputs of size n in *linear* worst-case comparison-time? On the other inputs, the algorithm is allowed to run arbitrarily slower. Justify your answer. If you state that such a comparison-based algorithm cannot exist, then prove why it can't exist. If you state that such an algorithm can exist, then provide the pseudo-code for such a sorting algorithm and clearly explain why it runs in linear time on 2^n of its input lists.

Question 4 [20 marks]
Recall that the zero-one principle states that comparison-based sorting algorithms that sort all *binary* lists, must sort all lists. The proof proceeds by contradiction, assuming that there is a list, say $L = (a_1, \ldots, a_n)$, that is not sorted by the algorithm. A binary list (i.e. a list for which all elements are zero or one) then is constructed from the sequence L. This binary list can be shown not to be sorted by the algorithm. We referred to such a binary list as a "binary-witness list." Consider a list $L = (a_1, \ldots, a_n)$. For each list element a_i of this list we define a function f_{a_i} in the same spirit as the function used to produce binary-witness lists:

$$\text{if } x \leq a_i \text{ then } f_{a_i}(x) = 0 \text{ otherwise } f_{a_i}(x) = 1$$

This function f_{a_i} can be applied to each element of L to produce the binary-witness list

$$(f_{a_i}(a_1), \ldots, f_{a_i}(a_n)).$$

In general you cannot reverse this process, i.e. it is not possible in general to recover the list $L = (a_1, \ldots, a_n)$ from a *single* binary-witness list $(f_{a_i}(a_1), \ldots, f_{a_i}(a_n))$. In other words, such a binary witness-list forms a "lossy encoding." Consider the *list* of binary-witness lists produced by the functions $f_{a_1}, f_{a_2}, \ldots, f_{a_n}$ when applied to the list $L = (a_1, \ldots, a_n)$:

$$(**)\ [(f_{a_1}(a_1), \ldots, f_{a_1}(a_n)), \ldots, (f_{a_n}(a_1), \ldots, f_{a_n}(a_n))]$$

Show that this list of binary-witness lists together with the *set* $\{a_1, \ldots, a_n\}$ of all elements contained in the list L is sufficient to recover the original list $L = (a_1, \ldots, a_n)$. Specify a decoding that computes L from this list of binary witness-lists and the set $\{a_1, \ldots, a_n\}$.

Hint for question 4: Consider a list $L' = (7, 1, 5, 3)$. Check how you can recover this list from the following *list* of binary witness-lists and where you can make use of the set of values $\{7, 1, 5, 3\}$ to achieve the decoding. The list of binary-witness lists in this case is:

$$[(f_7(7), f_7(1), f_7(5), f_7(3)),\ (f_1(7), f_1(1), f_1(5), f_1(3)),$$
$$(f_5(7), f_5(1), f_5(5), f_5(3)),\ (f_3(7), f_3(1), f_3(5), f_3(3))]$$

a) [4 marks] Compute the bit-values for each element of these binary-witness lists.

b) [8 marks] Explain how you can decode the original list from the given list of four binary-witness lists *and* from the set of values $\{7, 1, 5, 3\}$.

c) [8 marks] Generalise your approach obtained under part b) of this question to arbitrary lists L, i.e., explain how to decode such a list $L = (a_1, \ldots, a_n)$ from its set of values $\{a_1, \ldots, a_n\}$ and the *list* of binary-witness lists displayed in $(**)$ above.