Data structures and algorithms (2022/23) Written exam February 1st, 2023

This test must be taken individually. Any and all literature may be used while taking this test. In your answers be precise, and: (i) answer the questions *as they were asked*; and (ii) answer *all* tasks – if you will be answering to all tasks you might get bonus points.

Time: 105 minutes.

We wish you a lot of success – veliko uspeha! Veliko uspeha!

TASK	POINTS	OF POINTS	TASK	POINTS	OF POINTS
1			3		
2			4		

NAME AND SURNAME: ______________________

STUDENT ID: ______________________

DATE: ______________________

SIGNATURE: ______________________

Task 1. *Introduction and basics.* We have the following program:

```
foo(int array a) {
  int n= a.length; // array length
  int b= a[n-1];
  int j= 0;
  for int i= 0 to n - 2 {
    if a[i] < b {
      c= a[i]; a[i]= a[j];  a[j]= c;
      j= j + 1
    }
  }
  c= a[j]; a[j]= a[n]; a[n]= c;
}
```

QUESTIONS:

$\boxed{\textbf{1.}}$ Descriptively justify what the above program does.

$\boxed{\textbf{2.}}$ What is the invariance before the beginning of the inner for loop? Justify your answer.

$\boxed{\textbf{3.}}$ What is the time complexity of the program? Prove your answer.

Task 2. *AVL trees.* An AVL tree node is defined as (r, b, L, R), where r is the root value, b is the information about which subtree is higher (`0`, `L` or `R`) and L and R are both subtrees. An AVL tree is a dictionary implementation with all three functions.

QUESTIONS:

$\boxed{\textbf{1.}}$ (i.) Insert the integers $1, 2, \ldots 7$ in turn into the initially empty AVL tree. Draw a tree after each insertion. (ii.) Write an algorithm that will print out the vertices of an arbitrary AVL tree by layers.

$\boxed{\textbf{2.}}$ (i.) Extend the definition of an AVL node so that you can effectively answer the query `Number(T, v1, v2)` which returns a number of elements x in tree T such that $v_1 \leq x \leq v_2$. Such a dictionary (AVL tree) is called a *counting* dictionary (AVL tree). (ii.) Write an algorithm that executes a query and returns a number of elements in the counting AVL tree. (iii.) What is the time complexity of your algorithm? Justify the answer.

$\boxed{\textbf{3.}}$ (i.) Suppose we have an implemented countable AVL tree with a function `Number`. How can we directly use such an augmented tree to create a data structure that supports, in addition to the usual functions of a dictionary, also the functions `Even(T, v1, v2)` and `Odd(T, v1, v2)`, which return the number of even and odd elements between v_1 and v_2 in a tree T respectively? (ii.) What is the time complexity of your solution for all operations of a counting dictionary? Justify your answer.

HINT: Perhaps we could have several instances counting AVL dictionaries?

Task 3. *The Tribonacci numbers and dynamic programming.* On one hand, the Tribonacci numbers are a generalization of the Fibonacci numbers. The Tribonacci number $T(n)$ is defined as

$$T(n) = \begin{cases} 1 & n \leq 3 \\ T(n-1) + T(n-2) + T(n-3) & \text{otherwise.} \end{cases} \quad (1)$$

On the other hand, dynamic programming is a common method for solving the $0/1$ *knapsack problem*. In the latter case, we have n items, where the ith item has size s_i (`S[i]`) and value v_i (`V[i]`). Furthermore, we have a knapsack of size σ into which we put whole items so that the sum of their sizes is the largest, with their total size not exceeding the size σ of a knapsack. The question is what items should we put in the knapsack such that the maximum value is obtained. In this task, we will solve the problem with the following items:

i	1	2	3	4	5	6	7	8
s_i	1100	4000	1600	3200	4500	4800	900	4400
v_i	6	9	4	7	6	7	5	9.

(2)

QUESTIONS:

$\boxed{\textbf{1.}}$ (i.) Calculate $T(11)$ recursively. Show the calculation. (ii.) Write a recursive algorithm to compute $T(n)$ based on the definition (1) and with the use of memorization. (iii.) What is the time and space complexity of your algorithm? Justify the answer.

$\boxed{\textbf{2.}}$ (i.) We have a knapsack of size 1600 and 8 items in the table (2). Which items should we put in knapsack so that the sum of their values will be the largest? Justify the answer. (ii.) Write down the recursive definition of the $0/1$ knapsack problem (cf. the recursive definition of the Tribonacci numbers (1)) with knapsack size σ, and n items of sizes s_i and with values v_i ($1 \leq i \leq n$).

Task 4. *Graphs.* An undirected graph $G_C = (V_C, E_C)$ is a *clique* if $\forall u, v \in V_C : (u, v) \in E_C$. We denote a clique with $|V_C| = k$ as $G_{kC} = (V_{kC}, E_{kC})$. The second definition we need is a *subgraph.* A graph $G_P = (V_P, E_P)$ is a subgraph of a graph $G = (V, E)$ if $V_P \subseteq V$ and $E_P \subseteq E$. Here, nodes can be arbitrary relabelled.

QUESTIONS:

$\boxed{\textbf{1.}}$ (i.) Find the size $|E_{12C}|$ of E_{12C}? Justify the answer. (ii.) Write an algorithm that determines whether G_{3C} is a subgraph of a graph $G = (V, E)$. Let G be given by an adjacency matrix `A[u,v]`. (iii.) What is the time complexity of your algorithm? Justify your answer.

$\boxed{\textbf{2.}}$ Peter Puzzle would not be Peter if he did not find something interesting online. This time he found a function `Subgraph(G1, G2)` that determines if G_2 is a subgraph of G_1. The time complexity of this function is $T(n, k)$, where $|V_1| = n$ and $|V_2| = k$. Both G_1 and G_2 are given by an adjacency matrix.

(i.) Help Peter with an algorithm that uses the above function and determines if G contains G_{12C}. (ii.) What is the time complexity of your solution? Justify the answer.

$\boxed{\textbf{3.}}$ (i.) Write a *nondeterministic* algorithm that determines whether a clique G_{kC} is contained in G, where $|V| = n$. (ii.) What is the time complexity of your solution? Is it in NP? Justify the answer.