State Final Examination (Computer Science Sample Questions)

Summer 2021

1 Chomsky hierarchy (3 points)

Consider the language $L = \{w \mid w \in \{a, b\}^*, -1 \leq |w_a| - |w_b| \leq 1\}$ of words over the alphabet $\Sigma = \{a, b\}$, where the number of a's and the number of b's differ by at most 1. Classify the language L within the Chomsky hierarchy and find an automaton corresponding to the given class (finite, pushdown, linear bounded, Turing machine) accepting the language L.

Describe and name all the components of the definition of the automaton. A formal proof of the non-existence of an automaton of a lower class is not required.

2 Heaps (3 points)

1. Describe an implementation of a heap (as a data structure). Specifically, a binary min-heap.
2. Write pseudocode for the *ExtractMin* operation on your implementation of the heap.
3. Write an efficient pseudocode for the Heapsort algorithm. Specify and discuss its asymptotic complexity.

3 Relational databases (3 points)

Consider the following relational database schema:

Product(<u>ProductID</u>, ProductName, PriceCategory, Price)
Discount(<u>DiscountID</u>, DiscountName, NewPrice)
DiscountProduct(<u>DiscountID,ProductID</u>)
DiscountProduct.DiscountID $\subseteq$ *Discount*.DiscountID
DiscountProduct.ProductID $\subseteq$ *Product*.ProductID

Underlined are the keys of the relations. When more attributes are underlined together, this denotes a concatenated key.

1. Sketch the corresponding conceptual model (either a UML or an ER model).
2. Create SQL queries that return
   (a) all products that are not included in any discount,
   (b) all discounts that make some product more expensive than its standard price.
3. Decide if it would make sense to modify the schema above if you know that the *Product*.Price column functionally depends on the *Product*.PriceCategory column. If you think a modification should be made, describe why and how. If you think no modification should is necessary, explain why.

4 Tournament ranking (3 points)

Imagine you are designing a software for team tournaments that use the "all-play-all" system, that is, matches between all team pairs take place in an arbitrary order. In this assignment, we will focus on the part of the software responsible for determining the current standing of each team – specifically, at any point in the tournament, the software must compute a score for each team, based on the matches played so far. The score will be used to rank the teams.

Depending on the type of the tournament, the score can have multiple components – for example in soccer, the first component of the score can be the goal difference, the second component can be the yellow card count (simplified, real soccer tournaments use more complex scoring), each component is computed using a particular scoring criterion. The overall ranking process thus (1) uses a specific criterion to compute one score component for each team, match, and criterion, (2) adds up all score components for the same team and criterion, and (3) orders the teams lexicographically using the score component vector.

Choose one of C++, C#, or Java and use it to design the parts of the software identified below:

- The data structure used to represent a set of matches played – be aware that the information required for ranking depends on a particular sport (e.g., scores of sets in volleyball, overtime results in ice-hockey, or yellow cards in football). Try to make the system extensible (by extending the source code) to any team-based and match-based sport, without modification to the universal parts of the structure.

  Your design shall be guided solely by its use in rank determination. Persistent storage or program inputs are not part of your task.

- The data structure used to represent the tournament rules for the determination of the team ranking (i.e. their order). The rules are defined as a sequence of criteria - the first criterion is usually the number of points from the matches (note that there are different ways of assigning points), the subsequent criteria (applied when the previous criteria did not resolve the ranking completely) may be based on additional information stored with matches (e.g. goals scored, scores of sets, or yellow cards). All individual criteria are based on summing up the integer scores derived from individual match results and comparing the sums between teams. Show the interface of such a criterion and the class representing a sequence of such criteria.

  The set of available criteria must be easily extensible (by extending the source code) and a criterion implementation should be reusable for different tournaments, and, ideally, also for different sports whenever applicable (e.g., the goal difference for any sport based on goals). Any mechanisms common to all criteria (e.g. the summation over a set of matches) should not be included in the implementation of a criterion.

- Using the interfaces described above, provide a detailed implementation of the (reusable) goal-difference criterion and the (football-only) fewer-yellow-cards criterion, including the related parts of the match entry types.

5 Computer architecture and operating systems (3 points)

Consider the following class in pseudocode for C# and Java, comments show a C++ variant:

```
class Node {
    public:
    Node    next; // Node *next; for C++
    // some other data payload
};

class CountedList {
    public:
    CountedList() {
        root = null; // nullptr for C++
        count = 0;
    }

    int getCount() {
        return count;
    }

    void insertNode(Node n) { // (Node *n) for C++
        n.next = root; // n->next for C++
        root = n;
        ++count;
    }
    private:
    Node    root; // Node *root; for C++
    int     count;
};
```

Further, assume we have a multiprocessor system and the program starts multiple threads, which all have access to a shared variable of type `CountedList`. These threads will call the `getCount` and `insertNode` functions arbitrarily.

1. Is it possible that the *race condition* phenomenon occurs in the program? If so, write the line ranges or mark the lines directly in the code, that represent the critical section.
2. When calling `insertNode` from multiple threads, is it possible that the elements of the linked list will be correctly linked, but the number of elements in the `count` variable will be wrong? Elaborate.
3. If you believe that a race condition may occur in the code, try to fix the problem by editing the program (including adding new lines).

6 Routing (3 points)

Let us have a router performing standard routing and forwarding according to the following routing table:

Destination	Netmask	Interface	Gateway	Metric
195.113.19.0	255.255.255.0	(A) 195.113.19.20	on-link	1
172.217.23.0	255.255.255.0	(B) 172.217.23.55	on-link	1
216.58.0.0	255.255.0.0	(C) 216.58.201.78	on-link	1
185.17.100.0	255.255.255.0	(B) 172.217.23.55	(R) 172.217.23.111	10
185.17.200.0	255.255.255.0	(B) 172.217.23.55	(S) 172.217.23.222	10
104.0.0.0	255.0.0.0	(C) 216.58.201.78	(T) 216.58.90.100	10

In order to simplify the situation, the individual network interfaces and addresses of additional routers are denoted by letters. You also know that the corresponding MAC (EUI-48) addresses of these routers are the following: (R) FC-77-74-19-41-1E, (S) C8-F7-50-22-91-BA a (T) 00-15-5D-C7-B9-EA.

First, explain the meaning of the individual fields (columns) in the routing table.

We gradually receive the following standard IPv4 datagrams on the network interface (A) 195.113.19.20:

1. Datagram for recipient 185.17.200.50
2. Datagram for recipient 195.113.19.20
3. Datagram for recipient 74.6.231.21
4. Datagram for recipient 216.89.23.11
5. Datagram for recipient 255.255.255.255
6. Datagram for recipient 216.58.255.255

Describe how a given datagram will be processed in each of the previously enumerated situations. Explain and justify. If a given datagram will be forwarded, what IP address of the recipient will be filled in in the corresponding field (Destination Address) of the IP datagram header, and, analogously, what MAC (EUI-48) address of the recipient will be filled in in the corresponding L2 frame into which a given datagram will be inserted?

7 Optimization methods (3 points)

1. State the Minkowski-Weyl theorem for polytopes and provide an illustration of the theorem on a polytope with at least five vertices.
2. Given a bipartite graph $G = (U, V, E)$, write down the linear program describing the convex hull of all perfect matchings in G.
3. Prove that the linear program that you wrote indeed describes the required set.

8 Extremal graph theory (specialization question – 3 point)

1. State Turán's theorem and Ramsey's theorem.
2. Prove that for every positive integer k, there exists N such that the following claim holds. Consider a coloring of edges of the complete graph with $n \geq N$ vertices using three colors (red, green, blue) such that at most $n^2/\log n$ edges are green. Then there exists a set K of vertices of this graph of size k such that either all edges joining the vertices of K are red, or all these edges are blue.

9 Kempe chains (specialization question – 3 points)

1. What are Kempe chains?
2. Let v be a vertex of degree four in a graph G, and suppose that the neighbors of v induce a 4-cycle in G, and that the graph $G - v$ is 4-colorable. Prove that G is 4-colorable or G contains a subdivision of K_5 as a subgraph.

10 Set theory (specialization question – 3 points)

1. State Zorn's lemma (the maximum principle).
2. State the continuum hypothesis.
3. Which of the following statements can be proven in ZF (that is, without the axiom of choice)? Which of the following statements can be proven in ZFC (that is, with the axiom of choice)? For each of the questions it is sufficient to write a list of the statements, but you can also add a short explanation.
   (a) Zorn's lemma
   (b) Continuum hypothesis
   (c) Every set can be well-ordered.
   (d) Every countable set can be well-ordered.
   (e) Every finite set can be well-ordered.
   (f) The union of countably many countable sets is always countable.
   (g) If $f: a \to b$ is a surjective function, then there is an injective function $g: b \to a$ such that for every $y \in b$ we have $f(g(y)) = y$.
   (h) If $f: a \to b$ is an injective function and $g: b \to a$ is an injective function, then a and b have the same cardinality.

11 Morphological, Syntactic and Semantic Analysis of Natural Languages (specialization question – 3 points)

Lexical semantics studies the meaning of individual words or notions of natural languages. There are several basic approaches to the description of lexical meaning. Explain the following three methods of description of lexical properties of individual words and briefly describe their advantages and disadvantages:

1. Ontology
2. Semantic networks
3. Semantic features

12 Basic Formalisms for Description of a Natural Language (specialization question – 3 points)

Describe basic characteristics of following formalisms for the description of natural language syntax:

1. Transformational Grammar
2. Unification Grammar
3. Functional Generative Grammar

13 Basic Information Theory (specialization question – 3 points)

1. Draw a simple sketch of the entropy of a binary random variable. Describe and explain both the x-axis and the y-axis.
2. Draw a diagram that shows the relation between the entropies of two random variables X and Y and their mutual information, joint entropy and conditional entropies.