State Final Examination (Computer Science Sample Questions)

Fall 2020

1 Automata and grammars (3 points)

1. State the Pumping Lemma for Context-free Languages.
2. Consider the language $L = \{a^2b^ic^i \mid i \in \mathbb{N}_0\}$ (two a's, then any number of b's followed by an equal number of c's). Classify L within the Chomsky hierarchy. That is, determine the smallest class (highest type) of the Chomsky hierarchy the language L belongs to, prove that it belongs to that class and that it does not belong to a smaller class.

2 Topological sort (3 points)

1. Define a topological sort of a directed graph. Write a characterisation of graphs that can be topologically sorted.
2. Describe an algorithm constructing a topological sort of a directed graph.
3. Let us have a directed graph G which can be topologically sorted and two of its vertices u and v. We want to find *effectively* all vertices which lie on some directed path from u to v. Explain briefly the complexity of your algorithm (based on the number of vertices n and edges m of the graph G).

3 Databases (3 points)

1. What pairs of database operations in a transactional schedule are considered to be in conflict? Explain when two transactional schedules with the same set of transactions are conflict–equivalent.
2. Consider transactions T_1: $R(Y)$ $W(Y)$ $W(X)$ and T_2: $R(Y)$ $W(Y)$ $R(X)$ and the schedule S: $R_1(Y)$ $W_1(Y)$ $R_2(Y)$ $W_2(Y)$ $R_2(X)$ $W_1(X)$ $COMMIT_2$ $COMMIT_1$. Is the schedule S conflict–serializable? Explain, and, if appropriate, propose a modification to obtain a conflict–serializable schedule.
3. Consider relation *R(A,B,C), where* F=$\{A \to B, B \to C, C \to A\}$. Does this relation meet the requirements of the normal forms? If not, how would it be appropriate to change this data model fragment and why?

4 Object–oriented design (3 points)

Assume that we are designing a core of a web application framework – all of the code fragments below are written in the C# language, however, your solution can be written in C#, Java, or C++. For simplicity ignore any error states or unexpected argument values in your solution. You can implement any additional methods or types beyond the requirements of each of the following tasks.

1. We have the following `MasterController1` class that dispatches processing of incoming requests based on their `id`s:

   ```
   class MasterController1 { public string HandleReq(string id) { } }
   ```

   Every request type (with a unique `id`) should be processed by its own independent class, here denoted as request controller. Implement a `GreetingsController1` class for processing requests with ID "`welcome`" – it should always return a constant string "`Hello`"; then implement a `SumController1` class for "`sum`" requests that always returns a sum of integers 10 and 20 in textual representation as a string. Assume all instances of these classes are stateless, however, more complex processing of incoming requests can be added in the future.

   Complete the `MasterController1` class implementation so that it is possible to register "request ID" and "instance of a request processing class `C`" pairs with it in the following way:

```
var mc = new MasterController1();
mc.RegisterController("welcome", new GreetingsController1());
mc.RegisterController("sum", new SumController1());
```

Implement the `HandleReq` method of the `MasterController1` class so that for every request, it forwards its processing to the correct registered controller instance, identified by the request `id`, and returns the result of the controller's processing method.

2. Assume we now have a version V2 of:

   ```
   class MasterController2 { public string HandleReq(string id, string[] args) { } }
   ```

   In this new version we assume request controllers are stateful – i.e. processing of every request is done in several steps, where we pass individual arguments to the request controller, and only after that it generates the final request processing result as a single string value.

   Complete the `MasterController2` class implementation so that it passes the parameters from the `args` array, one after another, to the request controller selected using the request `id`. After that, the processing method of the request controller should be called and its result returned to caller. Also implement version V2 of the `GreetingsController2` class that returns text "`Hello`" followed by space–separated parameters passed to the controller; and version V2 of the `SumController2` class that takes the parameters passed to it as integer numbers and returns their integer sum in textual representation. The controller registration should remain the same as in task 1 – i.e. the following code using the `mc` variable constructed in the same way as in task 1, but with V2 versions of all the classes:

   ```
   L1: Console.WriteLine(mc.HandleReq("welcome", new[] { "Princess", "Consuela", "Banana", "Hammock" }));
   L2: Console.WriteLine(mc.HandleReq("sum", new[] { "1", "2", "39" }));
   L3: Console.WriteLine(mc.HandleReq("sum", new[] { "1000", "500" }));
   ```

   should print the following (note the last value is 1542 and not 1500, as the `MasterController2` class is repeatedly using the same registered instance of `SumController2` for every request with the "`sum`" ID):

   ```
   Hello Princess Consuela Banana Hammock
   42
   1542
   ```

3. Modify your solution of task 2 so that the `MasterController2` class will use a new instance of the request controller class (registered for the specific request ID) for every request. When running the code from task 2, the modified `MasterController2` should use one instance of the `SumController2` on line L2 (returning 42), and use a different instance of the `SumController2` on line L3 (which will then return 1500 instead of 1542).

   You will need to modify the registration process for the "request ID" and "request processing class `C`" pairs. Your implementation of the `MasterController2` needs to introduce an approach for passing a "recipe" or a "tool" to construct the `C` type instances.

5 Computer architectures: FAT (3 points)

We will try to work with the older FAT file system, specifically the FAT12 version (used on floppy disks). This file system stores data in blocks called *clusters*, which are identified by their sequence number. In addition to the data space, the file system stores on disk a *FAT table*, which contains one entry for each cluster (it is an array indexed by the cluster number, here 0-based). Each entry says whether the corresponding cluster is occupied by a file, and, if so, which cluster in the file comes next.

In the FAT12 format, each FAT table entry is 12 bits long, hence two entries in FAT12 are stored in three bytes. When these bytes are hexadecimal UV, WX, YZ, then the entries are XUV and YZW. The permitted entry values are:

Value	Meaning
000	Free cluster
002-fef	Cluster in use
ff0-fff	Cluster in use, end of file

This is a hexadecimal listing of the beginning of the FAT table (the rest of the table contains only 0):

```
000200   f0   ff   ff   00   40   00   05   60   00   07   80   00   09   a0   00   0b    ....@..`........
000210   c0   00   0d   10   01   ff   0f   01   ff   2f   01   13   40   01   15   60    ........./..@..`
000220   01   17   80   01   19   a0   01   1b   c0   01   1d   e0   01   1f   00   02    ................
000230   21   20   02   ff   0f   00   00   00   00   00   00   00   00   00   00   00  !  ..............
```

The root directory has only 3 entries captured in the following hexadecimal listing:

```
002600   46   4f   4c   44   45   52   20   20   20   20   20   20   00   00   00   00   FOLDER      ....
002610   00   00   00   00   00   00   a0   79   79   4f   03   00   9d   39   00   00   .......yyO...9..
002620   46   49   4c   45   20   20   20   20   54   58   54   10   00   00   00   00   FILE    TXT.....
002630   00   00   00   00   00   00   85   78   1f   51   0e   00   00   00   00   00   .......x.Q......
002640   4c   4f   43   43   4f   4e   46   20   58   4d   4c   20   00   00   00   00   LOCCONF XML ....
002650   00   00   00   00   00   00   dc   aa   6f   4c   0f   00   96   03   00   00   ........oL......
```

The directory entry has the following structure:

Bytes	Content
0-10	File name (8B) and extension (3B)
11	File attributes (bitvector):

Bit	Meaning
0	read only
1	hidden
2	system file
3	volume label
4	subdirectory
5	archive

12-21	Reserved
22-23	Time of last change (complex format, not essential)
24-25	Date of last change (complex format, not essential)
26-27	Starting cluster
28-31	Filesize in bytes

The FAT file system stores numeric values as little endian, bit 0 is LSB.

1. List the contents of the root directory (file name, whether it is a file or a directory, for files the size in bytes in decimal).
2. What clusters do you need to read if you read the LOCCONF.XML file sequentially?
3. Where and what (conceptually, it is not necessary to list the exact individual bytes) do I have to write if I append 1 KiB of data to the end of the LOCCONF.XML file?

6 Transport protocols and their properties (3 points)

Explain the following terms and briefly enumerate what we want to achieve through them and what problems we have to address to guarantee them.

1. Stream-oriented transfer
2. Connection-oriented transfer
3. Reliable transfer

Propose and carefully describe how you would guarantee these features in a hypothetical (yet meaningfully and effectively operating) transport layer protocol that is to work over the IP protocol in the network layer. You can get an inspiration from the existing protocols in the TCP/IP architecture, or you can design your own approach. In particular, address the following areas:

1. Creating an illusion of a stream transfer
2. Establishing a new connection
3. Ensuring reliability

7 Optimization methods, polyhedra (3 points)

1. Consider the polyhedron $P = \{x \in R^3 \mid x_1 + x_2 \geq 1,\ x_2 + x_3 \geq 1,\ x_1 + x_3 \geq 1\}$. Decide whether P is full dimensional and whether it is integral (i.e., all has vertices have all coordinates integral).
2. For a given graph $G = (V, E)$ consider the linear program $\min \sum_{e \in E} x_e$ s.t. $\sum_{e \in T} x_e \geq 1$ for each spanning tree T of the graph G, $x \geq 0$. What does an optimal integral solution to this LP correspond to?
3. Is the LP from the previous question integral for every graph G?

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

A systematic approach to morphology of natural languages can be based upon a variety of basic principles. There are three most frequently used methods of describing and processing morphology. They are based upon different basic morphological units, namely upon morphemes, lexemes or words. Explain briefly how each of the three methods works and for each of them give an example of a language type for which it is suitable.

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

Valency is one of the fundamental concepts of the theory of Functional Generative Description.

1. Explain the main difference between inner participants and free modifiers.
2. Name at least 3 out of 5 basic types of inner participants existing in the Functional Generative Description theory.
3. Define a notion of a valency frame. What does it contain?

10 Basic Information Theory (specialization question – 3 points)

We roll two dice, X and Y. The first dice, X, is perfect, i.e. the distribution of the values $\{1, 2, 3, 4, 5, 6\}$ is uniform.

The other dice, Y, is false so that the even values are twice as likely as the odd ones:

$$\Pr\{1\} = \Pr\{3\} = \Pr\{5\} \text{ and } \Pr\{2\} = \Pr\{4\} = \Pr\{6\} \text{ and } \Pr\{2\} = 2 \cdot \Pr\{1\}.$$

We consider the numbers from each toss as observations of random variables X and Y. Using dice Y, we also consider a binary random variable Y_b with two possible values, *even* and *odd*. The distributions of the variables are summarized in the following table:

random variable	**range**	**distribution**
X	$\{1, 2, 3, 4, 5, 6\}$	uniform
Y	$\{1, 2, 3, 4, 5, 6\}$	non–uniform
Y_b	$\{even, odd\}$	non–uniform

1. What is the unit of entropy? What is the unit of *conditional* entropy?
2. Are the variables X and Y statistically independent? Justify your answer.
3. What can we say about the entropy $\mathrm{H}(Y_b)$? Choose one of the following options and explain your answer.

   a) $\mathrm{H}(Y_b) > 2$ b) $\mathrm{H}(Y_b) < 2$ c) $\mathrm{H}(Y_b) = 2$