State Final Examination (Computer Science Sample Questions)

Spring 2023

1 Concatenation of palindromes (3 points)

1. Give formal definitions of all types of grammars in the Chomsky hierarchy.
2. A *palindrome* is a word that reads the same backwards as forwards (that is, a word that satisfies $w = w^R$). Let L be a language consisting of all words over the alphabet $\{a, b\}$ that are a concatenation of some pair of (not necessarily distinct) palindromes. Classify the language L within the Chomsky hierarchy and construct a grammar of the corresponding type that generates it. (You do not need to show that the language is generated by the grammar or that no grammar of a higher type exists.)

2 Sorting (3 points)

1. Describe an algorithm that sorts (i.e., permutes to a non-decreasing order) a sequence $x_1, \ldots, x_n$ of pairwise distinct items using binary search trees.
2. What changes if we allow multiple occurrences of the same item?
3. What conclusion can we draw about the minimum possible time complexity of operations with a binary search tree? You can assume that keys in the tree can only be compared.
4. What is the time complexity of your algorithm from subtask 1 if we use an AVL-tree and items will be strings of length L?

3 Databases (3 points)

1. Consider transactions T_1: $W(A)$ $R(B)$ $COMMIT$ and T_2: $W(B)$ $W(C)$ $COMMIT$. Is the schedule S: $W_1(A)$ $W_2(B)$ $R_1(B)$ $COMMIT_1$ $W_2(C)$ $COMMIT_2$ recoverable? If so, why? If not, how would it need to be modified to make it recoverable? Operation R represents the reading of the given variable, operation W represents its writing.
2. Describe the conditions necessary for a relational schema to be in the third normal form. Give a simple example of a relation that would not be in the given normal norm.
3. How does the binary relationship between entities *Person* and *Phone* differ on a conceptual level – if at all – if it is represented by the relation $PersonPhone(\underline{PersonID}, \underline{PhoneID})$ in one case, and by the relation $PersonPhone(\underline{PersonID}, \underline{PhoneID})$ in the other case – that is, with a single two-column key in the former and two independent single-column keys in the latter. The underscore specifies the key(s) of the relation.

4 Object-oriented design (3 points)

Assume we are implementing a bitmap image processing application. The user should be able to have multiple images open at once. The user opens a new image in the application's user interface (UI) by selecting its file name – the application decides on image format based on the file extension. With an open image, the user can gradually select some of the filters offered – in the application UI, the user always selects a filter and its specific settings, and it is then saved in a list of selected filters. At the end, the user will have the option to apply the selected filters together. Each open file can be saved by the user at any time either in the original or another supported bitmap format. For sake of simplicity, assume that one image format corresponds to exactly one file extension (i.e. for JPEG format only the extension `.jpg` is valid, and additional `.jpeg` is not), and that all filters can freely change the image data, but the pixel dimensions of the image are always preserved.

We have prepared a skeleton of the main classes of our program in C# (see below), where the `Image` class represents the loaded image data (we will remember other metadata like the original file name etc. elsewhere; note: `[,]` in C# means a two-dimensional array), and the `Program.Main` method represents a simple scenario of using the other classes:

```
struct Color { public byte R, G, B; }

static class Filters {
	public static void ApplySharpness(Image image, float amount, float radius) { ... }
	public static void ApplyBrightness(Image image, float amount) { ... }
}

abstract class Image {
	public Color[,] bitmap = null;
	public abstract void Load(string fileName);
	public abstract void Save(string fileName);
}

class PngImage : Image {
	public override void Load(string fileName) { ... }
	public override void Save(string fileName) { ... }
}

class JpegImage : Image {
	public override void Load(string fileName) { ... }
	public override void Save(string fileName) { ... }
}

static class ImageLoader {
	public static Image Load(string fileName) {
		Image image;
		if (fileName.EndsWith(".png")) {
			image = new PngImage();
		} else if (fileName.EndsWith(".jpg")) {
			image = new JpegImage();
		} else {
			throw new ArgumentException();
		}
		image.Load(fileName);
		return image;
	}
}

class Program {
	public static void Main() {
		var fileName = "a.jpg";
		var image = ImageLoader.Load(fileName);
		Filters.ApplySharpness(image, 9.0f, 1.3f);
		Filters.ApplyBrightness(image, 1.7f);
		image.Save(fileName);
	}
}
```

Reimplement the entire object design to better support implementing, maintaining, and further extending the application described above. Write the new code skeleton in C#, C++, or Java. Focus on the design of types and their relationships (write everything in the syntax of the chosen language), declaration of methods, and only key properties and fields. You do not have to provide an implementation for most methods – just edit the `Program.Main` implementation to make an appropriate use of your OO design. In your solution, please also implement the code that will represent the sequence of steps in your design from having the image file name to selecting the file format and loading the image in that format.

In your proposal, consider not only the easy possibility of implementing all aspects of the application usage scenario outlined above, but also the fact that in the future we want to add a large number of different filters – but we always plan to add the implementation of a new filter to the main source code of the application, we do not want to ever support new filters delivered as plugins. Therefore, in your solution, consider only the batch application of selected filters; you do not have to deal with any infrastructure for general filter description.

In contrast, we want to add support for new image formats using plugins. Assume that we will implement each newly

supported image format in the form of a new plugin to our application (which we'll load as a dynamically linked library at runtime) – so we want the process of loading images from a file to be flexible enough to allow us to provide a reference to a new file format that a plugin will provide (do not deal with actual plugin loading, or support for loading dynamically linked libraries, or directly searching for a description of the format in the plugin).

5 Process memory organization (3 points)

In both parts, assume the context of a common desktop operating system system and a process created by running an application written in C++, C# or Java.

Part A

1. The address space of each process is divided into different areas containing program code and data. What are the main areas where data (with different lifetimes and methods for allocation and deallocation of storage space) can be found in the process address space?
2. For each such area, explain what data is typically stored there, when storage for the data in the area is allocated and deallocated, and who is responsible for doing it. Include a code fragment (in C++, C# or Java) illustrating the allocation and deallocation in your explanation.

Part B

Assume that the language of the pseudocode shown below supports the concept of pointers and an operator that, when applied to a variable, provides the address at which the value of the variable is stored. Specifically, if `T` is a variable type, then `T*` represents the type *pointer to a value of type `T`* (in Pascal, the same concept is represented as `^T`). To obtain the address of a variable, we must apply the `&` operator to it. Given a variable `var` of type `T`, the result of `&var` is a typed pointer (i.e., a value of type `T*`), representing the address of the variable `var` (in Pascal, the `@` operator serves the same purpose).

For example, an integer variable `number` of type `int` with a value of `42` is defined as `int number = 42`. A variable `number_ptr` of type *pointer to type `int`* which contains the address of the variable `number` is then defined as `int* number_ptr = &number`.

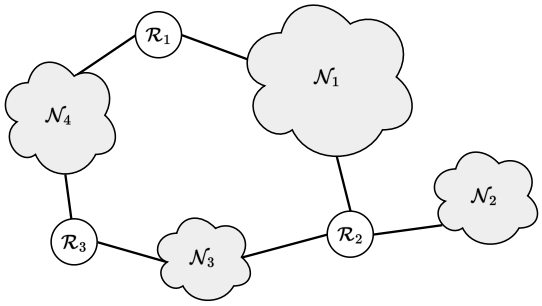
In the following program, the function `run()` displays a sequence of different addresses for each of the three blocks of code A, B and C (marked by comments). Assume that the function was called from the main thread, and that no other threads are running. Also assume that the code runs as written, i.e. the compiler did not optimize the code in any way. Assume common data type sizes, i.e., 4 bytes for `int` and 8 bytes for `long` and pointer types.

```
1  void descend(int steps) {
2      int level = steps;
3      print_pointer(&level);
4
5      if (steps > 0) {
6          descend(steps - 1);
7      }
8  }
9
10 class Record {
11     int value1;
12     long value2;
13 }
14
15 Record* records[100];
16
17 void run() {
18     // Block A
19     descend(100);
20
21     // Block B
22     for (int i = 0; i < records.length; i++) {
23        Record* record = new Record();
24        print_pointer(record);
25        records[i] = record;
26     }
27
28     // Block C
29     for (int i = 0; i < records.length; i++) {
30         print_pointer(&(records[i]));
31     }
32 }
```

For each of the blocks A, B and C, characterize the values of addresses that the program could display. The actual values are not important — focus on what values the addresses will be divisible by, whether the sequence of addresses is going to be increasing or decreasing, how large will be the differences between consecutive addresses, and what must be the minimal distance between addresses within each block. **Justify the characterization of each block of addresses!**

6 Routing (3 points)

Let us have a system of networks $\mathcal{N}_1$ to $\mathcal{N}_4$ interconnected by routers $\mathcal{R}_1$ to $\mathcal{R}_3$ with the topology shown in the following figure.



Routing tables of individual routers are defined as follows:

$\mathcal{R}_1$

Destination	Netmask	Interface	Gateway	Metric
210.18.76.0	255.255.255.0	(A) 210.18.76.101	on-link	1
195.113.0.0	255.255.0.0	(B) 195.113.19.20	on-link	1
210.18.0.0	255.255.0.0	(A) 210.18.76.101	(G) 210.18.76.202	10

$\mathcal{R}_2$

195.113.0.0	255.255.0.0	(C) 195.113.19.30	on-link	1
210.18.25.0	255.255.255.0	(D) 210.18.25.10	on-link	1
210.18.35.0	255.255.255.0	(E) 210.18.35.88	on-link	1

$\mathcal{R}_3$

210.18.35.0	255.255.255.0	(F) 210.18.35.99	on-link	1
210.18.76.0	255.255.255.0	(G) 210.18.76.202	on-link	1
default		(F) 210.18.35.99	(E) 210.18.35.88	20

End nodes in network $\mathcal{N}_1$ forward all non-local traffic to router $\mathcal{R}_1$, analogously in network $\mathcal{N}_2$ to $\mathcal{R}_2$, in network $\mathcal{N}_3$ to $\mathcal{R}_3$ and finally in network $\mathcal{N}_4$ also to $\mathcal{R}_3$.

Assume the following IPv4 datagrams are sent:

1. Datagram from (H) 210.18.35.152 for (I) 210.18.76.18
2. Datagram from (I) 210.18.76.18 for (J) 210.18.76.65
3. Datagram from (K) 195.113.45.17 for (L) 210.18.25.74
4. Datagram from (L) 210.18.25.74 for (J) 210.18.76.65
5. Datagram from (J) 210.18.76.65 for (A) 210.18.76.101
6. Datagram from (H) 210.18.35.152 for (M) 255.255.255.255
7. Datagram from (I) 210.18.76.18 for (N) 210.18.95.38
8. Datagram from (H) 210.18.35.152 for (O) 210.18.25.255

To simplify our work, let us symbolically denote all the used IP addresses and their corresponding unicast MAC addresses with letters (A) to (O).

For each of the previous situations, determine how a given datagram will be delivered from the sender to the intended recipient. That is, through what networks, what nodes, what interfaces and to what addresses. Explain and justify.

7 Basic Information Theory (specialization question – 3 points)

Let X and Y be two random discrete variables and H stands for Entropy. For the relations below write **true** or **false** for any X and Y. If it is **true**, prove it. If it is **false**, give a counterexample or prove that the opposite is true. Hint: use the relationship between Mutual information and Entropy.

1. $H(X) \geq H(X|Y)$
2. $H(X) + H(Y) \leq H(X, Y)$

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

1. Explain the following terms: *morphological analysis*, *lemmatization* and *tagging*.

2. Illustrate these procedures using some examples and use the Prague Dependency Treebank's annotation scheme.

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

1. Explain the term *treebank*. Name two treebanks that you are familiar with.
2. Describe non-projective dependency trees and give an example of non-projective dependency tree.